Querying existential rule knowledge bases : decidability and complexity

Swan Rocher

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de

Docteur

Délivré par l'**Université de Montpellier**

Préparée au sein de l'école doctorale **I2S**
Et de l'unité de recherche **UMR 5506**

Spécialité: **Informatique**

Présentée par **Swan Rocher**

Querying Existential Rule Knowledge Bases: Decidability and Complexity

Soutenue le 25 novembre 2016 devant le jury composé de :

Directrice de thèse		
Mme. Marie-Laure Mugnier	Professeur	Univ. de Montpellier
Co-encadrant		
M. Jean-François Baget	Chargé de Recherche	INRIA
Rapporteurs		
Mme. Marie-Christine Rousset	Professeur	Univ. de Grenoble
M. Sebastian Rudolph	Professor	Univ. Dresden
Examinateurs		
M. Christophe Paul	Directeur de Recherche	CNRS
M. Andreas Pieris	Lecturer	Univ. of Edinburgh

Université de Montpellier

Remerciements

Trois ans, c'est court, et pourtant c'est en écrivant ces remerciements que je me rends compte que j'ai eu l'occasion de rencontrer énormément de gens, qui m'ont tous apporté quelque chose plus ou moins directement, si bien qu'il est difficile d'être sûr de n'oublier personne...

Merci tout d'abord à Jean-François et Marie-Laure pour avoir été des directeurs de thèse parfaits ! Tant au niveau scientifique que personnel, vous avez toujours été là lorsque j'en avais besoin.

Bien sûr, merci aussi à Marie-Christine Rousset et Sebastian Rudolph pour votre lecture si attentive de ce manuscrit : ma thèse ne serait pas ce qu'elle est sans vos remarques. Merci également à Andreas Pieris et Christophe Paul pour avoir accepté d'être examinateurs, et merci aussi à Christophe pour avoir participé à mon comité de suivi de thèse ces trois ans !

On s'éloigne un petit peu, mais pas trop, merci à Michel et le projet Qualinca, sans qui je n'aurai pas pu faire cette thèse. Merci Annie pour avoir réussi à garder ton calme malgré ma hantise des démarches administratives. Et merci à toute l'équipe, ses permanents, ses stagiaires et bien sûr ses doctorants (passés et actuels), Michaël pour tes conjectures et tes contre-exemples, Stathis pour tes questions existentielles, Léa pour tes histoires à base de chats et/ou lapins, et tous les autres !

Non doctorant, mais c'est comme si, merci beaucoup Clément, co-bureau à travers les ages, pour les discussions du café, et pour tout le reste d'ailleurs !

On peut porter le regard un peu plus loin car le LIRMM n'est pas seulement une équipe. Merci à tous les gens avec qui j'ai pu discuter au détour d'un couloir ou autour d'un café. En particulier, merci beaucoup à Sabrina, Anaël, Guilhem, Valentin, Julien, François, Florian, Jessie et Nam' à la fois pour la détente, et pour les moments scientifiques. Ces années n'auraient certainement pas été les mêmes sans votre présence ! Non, je ne t'oublie pas, merci Chloé pour m'avoir aidé à en arriver là ! Et merci Marthe pour ton thé parfaitement équilibré et tes nombreux gâteaux !

Finalement, le LIRMM ne serait pas ce qu'il est sans Laurie et Nicolas : toujours souriants, et on ne peut plus efficaces, merci beaucoup pour tout ce que vous avez fait !

Plus loin que Montpellier (beaucoup), merci à Charlotte et Julien pour m'avoir accueilli dans un super paysage quand il fallait que je me change les idées ! Je n'oublierai pas mon "bureau" à la montagne.

Enfin, il est plus que temps de remercier la famille : merci Luc et Sarah pour votre soutien et n'avoir jamais douté. Et bien entendu merci pour tout Eva ! Il me semble aussi important de remercier la famille un peu moins directe : alors, merci Raphy pour ton accueil, et aussi pour la force de volonté que tu m'as apporté d'ailleurs. Merci également à Lyes, malgré tes blagues douteuses, tu m'as permis de rigoler quand je ne m'y attendais pas ! Et évidemment, merci à Sandie pour m'avoir supporté pendant la rédaction de cette thèse !

En bref, merci à tout le monde !

Contents

Introduction

Querying Knowledge Bases

The recent years have been marked by a tremendous increase of the volume and the heterogeneity of available data sometimes referred to as the "data deluge". Exploiting these data has become a major issue in several research domains (knowledge representation and reasoning, data management, Semantic Web, ...) and the need for integrating data semantics into querying mechanisms has been widely acknowledged. This has renewed the interest for *ontologies*, which are typically used to formalise general background knowledge on the modeled domains.

Indeed, ontologies have several qualities with respect to better exploiting data:

- they can be used to integrate heterogeneous data from different sources by providing a common vocabulary;
- they allow to adapt the querying vocabulary to specific users' needs, hence abstracting from how data are actually stored;
- they allow to infer knowledge that is not explicitly stored in the data, hence palliating incompleteness in the data.

However, taking into account data semantics requires both suitable ontological formalisms and new query answering mechanisms able to integrate ontological knowledge. Indeed, classical query answering mechanisms have been tailored and optimised for databases. Now, the focus has been shifted to *knowledge bases*, in which an ontological layer is added on top of data.

This motivated a new research line that led to proposals known as *Ontology-Based Data Access* (OBDA, e.g., [CDL+07, PLC+08]), *Ontological Query Answering* (e.g., [CGP11, Mug11]) or *Ontology-Mediated Query Answering* (OMQA, e.g., [BO15, Bie16]). In these approaches, the ontological layer is seen as a logical theory in (a fragment of) first-order logic and data are abstracted into logical facts (which can be mapped to actual data).

Existential Rules

These new issues have deeply influenced research in Description Logics, the major family of formalisms to represent and reason with ontologies [BCM+03]. This led

to the definition of new description logics, generally called lightweight description logics, such as the DL-Lite [CDL+07] and the $\mathcal{EL}$ [BBL05] families, as well as the associated profiles of the Semantic Web language OWL 2. At the same time, a new logical framework has emerged, called *existential rules* [BLMS11, KR11], also known as Datalog+/- [CGL12]. Existential rules have a double origin: on the one hand, they were designed as an extension of Datalog, the language of deductive databases [AHV95], for ontological representation purposes, hence the name Datalog+/-; on the other hand they correspond to the logical translation of rules in a graph-based knowledge representation framework [CM09]. They also have the same form as high-level constraints from the database theory, known as Tuple Generating Dependencies (TGDs) [BV84]; note however that TGDs define constraints on the data, while existential rules are used to infer knowledge on the data. Finally, it appears that the existential rule framework generalises most of the new description logics developed for querying data.

More precisely, existential rules are positive and conjunctive rules of the form "if *body* then *head*", with a specific feature: it is possible to introduce in the head of a rule variables that do not occur in the body. These variables are existentially quantified. Thanks to this particularity, existential rules are able to infer the existence of individuals not necessarily present in the initial data. This makes them well-adapted to open-world reasoning, in which not only we cannot assume that only what is explicitly encoded in the data is true, but also in which we cannot suppose that the only known objects are those present in the data. Such a feature is considered important for representing and reasoning with ontological knowledge.

Let us illustrate this with a simple example. Consider two roommates Bob and John, and assume we know that John pays for Internet. We can see these data as a set of facts $roommates(Bob, John) \wedge paysInternet(John)$. We now want to know if Bob has Internet at home and so we ask the query $\exists x(livesIn(Bob, x) \wedge hasInternet(x))$. Without ontological knowledge we are not able to answer positively. Despite, we intuitively would like to answer yes: some ontological knowledge is missing to get a positive answer. First, we know that if two persons are roommates, there is some place where they both live, hence the rule $\forall x \forall y(roommates(x, y) \rightarrow \exists z(livesIn(x, z) \wedge livesIn(y, z)))$. We also want to say that if someone pays for Internet and lives somewhere, then Internet is present in that "somewhere", which we formalise by the rule $\forall x \forall y(livesIn(x, y) \wedge paysInternet(x) \rightarrow hasInternet(y))$. Now our query gets a positive answer, indeed Bob and John, being roommates, live in the same place. John pays for Internet, hence he gets Internet at home. Therefore, Bob lives in a place where there is Internet. We could even add some other rules to complete the background knowledge of this example, for instance, we could say that roommate is a symmetrical relation, which we could express with the rule $\forall x \forall y(roommates(x, y) \rightarrow roommates(y, x))$.

Existential rules have the simplicity of rule-based languages, a privileged form to express human knowledge, as well as their flexibility, i.e., they are able to adapt to various kinds of data and are easily extended to encode new information. While

being simple, they are highly expressive, which allows to encode various kinds of ontological knowledge. Unsurprisingly, this expressivity has a cost: indeed, most reasoning problems over existential rules knowledge bases are undecidable (e.g., from [BV81] on TGDs). This motivated intense research in the last years to find classes of rules for which reasoning is decidable and hopefully tractable. Currently, a large landscape of decidable classes is known with different expressivity-complexity trade-offs.

This thesis makes further contributions to the study of decidable existential rule classes and to the analysis of their complexities.

Contributions of this Thesis

We consider knowledge bases in which the ontology is a set of existential rules. About the queries, we consider conjunctive queries, which can be seen as existentially quantified conjunctions of atoms. These queries correspond to the basic and most used queries in databases. Hence, the fundamental problem we study is the *conjunctive query entailment problem (CQ entailment)*, which asks whether a conjunctive query is entailed by a knowledge base.

Our contribution is twofold. First, we analyse the different "acyclicity-based" decidable rule classes found in the literature. These rule classes rely on acyclicity conditions to ensure the finiteness of some forward chaining algorithm. We propose a tool that allows us to unify most acyclicity notions and to extend them in generic way. We also analyse the complexity of the recognition problem, i.e., deciding whether a given set of existential rules satisfies these new acyclicity notions. The main paper associated with this work is [BGMR14a].

Second, we consider the decidability (and complexity) of the CQ entailment problem when combining known decidable rule classes with a frequently required construct in ontological modeling, namely the transitivity of binary relations. We clarify the decidability picture for all the (currently known) classes for which some forward chaining algorithm halts. Then, we study the particular case of linear existential rules, and show that, up to a minor safety condition, linear rules are compatible with transitivity. We finally consider the complexity of the CQ entailment problem over knowledge bases composed of (safe) linear and transitivity rules. Most of this work is reported in [BBMR15], however, with respect to that paper, we provide a new undecidability result and correct a flaw in a complexity proof.

Other Contributions

During these three years, we have also considered other issues related to existential rules. The results obtained are not detailed in this thesis, hence we briefly mention them below, and refer the interested reader to the associated papers.

Our work on acyclicity-based decidable classes of rules was also motivated by the objective of extending decidable cases of CQ entailment for existential rules extended with a particular nonmonotonic negation, namely *stable negation*. Briefly, adding nonmonotonic negation to existential rules allows to apply rules only if some negated atoms from the rule body are not entailed by the knowledge base. While it is easy to see that stable negation may complicate the design of algorithms and increase the complexity of the CQ entailment problem, it also appears that it may make this problem undecidable, even when stable negation is added to a decidable class of positive existential rules. We have thus proposed a way to extend the results obtained for acyclicity-based positive existential rules to existential rules with stable negation. This work has been published in [BGMR14b].

Another interesting issue is how to deal with inconsistent knowledge bases. In this thesis, we assume that knowledge bases are consistent, but in practice, there are reasons to believe that it is not a safe assumption. Indeed, in a world where such a volume of data is available, which furthermore come from different data sources, it is likely that inconsistencies arise. From a logical point of view, an inconsistent knowledge base allows to infer everything, and in the context of querying data, it is obviously not the desired behaviour. We focus on the case where the ontology is assumed to be consistent, hence inconsistencies in the knowledge base come from the data (which is the most considered setting). When it is not possible to effectively repair the data, one has to consider query mechanisms tolerant to inconsistencies. Various inconsistency-tolerant semantics have been defined, which all rely on a common idea: answers to queries are drawn from some maximally consistent subsets of the data (possibly added with some consistent inferred knowledge), called repairs. We have taken part in the definition of a general framework that unifies most inconsistency-tolerant semantics from the literature [BBB$^+$16a]. Briefly, this framework sees an inconsistency-tolerant semantics as a pair, composed of a "modifier" of the inconsistent knowledge base that allows to select some maximal repairs, and an "inference-strategy" that allows to infer conclusions from these selected repairs. Concerning the notion of maximal consistent subset, at least two different measures can be interesting: maximality in terms of set inclusion and maximality in terms of cardinality. The inference strategy may want to find an answer in all selected repairs, in their intersection, in a single one, or in a majority of them. These choices give rise to different semantics, some of them corresponding to existing proposals, while others are new. Our specific contribution in the context of this general framework consisted in analysing the data complexity of the obtained semantics for the *fus* subset of existential rules, which generalises most members of the DL-Lite family [BBB$^+$16b].

Finally, we have also developed Kiabora, a software tool that allows to recognise most known decidable rule classes and proposes ways of combining them [LMR13]. The first version of this tool is available via a web interface at `www.lirmm.fr/`

`kiabora`. The second version has been integrated into the toolkit Graal dedicated to querying knowledge bases within the existential rule framework. Graal provides an abstract layer that allows to store and query various kinds of data (relational databases, RDF triple stores, internal memory, ...), forward and backward chaining algorithms, as well as complementary tools such as Kiabora and translators from or to other languages. More information can be found on Graal website at `http://graphik-team.github.io/graal/`. Beside the development of Kiabora itself, we have helped to design and implement part of Graal [BLM+15, BGL+15].

Organisation of the Thesis

The remainder of this thesis is organised in as follows.

In **Chapter 1**, we define required mathematical notions and introduce the existential rule framework. We recall various well-known results that we will use throughout the manuscript.

In **Chapter 2**, we present an overview of the main rule classes for which the CQ entailment problem is decidable. We first recall three abstract rule classes that ensure a specific behaviour of some reasoning algorithms. These classes are abstract in the sense that the associated recognition problem is undecidable. Then we review the main known concrete decidable rule classes, which are obtained by enforcing some syntactic restrictions on the sets of rules. Most of them belong to one or more abstract rule classes.

Chapter 3 is devoted to decidable rule classes that rely on acyclicity conditions ensuring the halting of some forward chaining variant, also known as chase in database theory. The first part of this chapter is dedicated to the main known chase variants, for which we give a unified formal definition. We then detail acyclicity-based rule classes, which rely on different graphs. We propose a way to unify these different acyclicity notions with a new graph and associated notion of dangerous cycles. Finally, thanks to the tool we used to unify them, we extend previous acyclicity-based rule classes.

In **Chapter 4**, we are interested in combining the transitivity of binary relations with known decidable rule classes. We first review known results on this issue obtained in the context of existential rules. Then, we provide new undecidability results, which allows to clarify the picture for rule classes for which some forward chaining algorithm halts. Finally, we consider the case of linear existential rules, one of the simplest, yet useful, classes of existential rules. We show that, up to a minor safety condition, linear and transitivity rules are compatible, and provide complexity results for the CQ entailment problem in terms of data and combined complexity.

Finally we summarise our contributions and outline further research in the conclusion.

Chapter 1

Fundamental Notions

Contents

1.1 General Mathematical Notions

In this section we define several basic mathematical notions and notations needed in this thesis.

1.1.1 Basic Notations

Given a function f, we denote by $dom(f)$ the domain of f. If we are also given a set X, we denote the *restriction* of f to X by $f|_X = \{(x, f(x)) \mid x \in dom(f) \cap X\}$.

A *sequence* S is a function whose domain is $\mathbb{N}$ if it is infinite, $\emptyset$ if it is of length 0, and $\{0, \ldots, n-1\}$ if it is of length $n \in \mathbb{N}^+$. We denote by S_i the i^{th} element of S, i.e., $S_i = S(i)$.

1.1.2 Graph Notions

Given a (hyper)graph G, we denote by $V(G)$ its set of vertices and $E(G)$ its set of (hyper)edges. A (hyper)graph can be either directed or undirected, in the former case edges are ordered, while in the latter they are not, i.e., in an undirected (hyper)graph, edges (x, y) and (y, x) are the same object.

Given a directed graph G and a vertex $v \in V(G)$ we denote its neighbourhood in G by $\Gamma(v) = \{x \mid (v, x) \in E(G)\}$.

Definition 1.1 (Path)
Given a graph G, a path *is a sequence of edges $(x_1, x_2), \ldots, (x_{k-1}, x_k)$ such that for all $1 \leq i < k$, $(x_i, x_{i+1}) \in E(G)$.*

If G is directed we say that the path is from x_1 to x_k *(and we say that x_k is* reachable *from x_1). If G is undirected we say that the path is* between x_1 *and x_k.*

Now can be defined the notion of a connected component in an undirected graph.

Definition 1.2 (Connected Component)
Given an undirected graph G, a connected component *of G is a subset $C \subseteq V(G)$ such that for any $u, v \in C$, there is a path between u and v.*

This notion has been extended to strongly connected components for directed graphs as follows.

Definition 1.3 (Strongly Connected Component)
Given a directed graph G, a strongly connected component *of G is a subset $C \subseteq V(G)$ such that for any $u, v \in C$ there is a path from u to v (i.e., v is reachable from u).*

In Chapter 2, we also use the notion of treewidth of a (hyper)graph. This notion relies on the notion of tree decomposition defined next.

Definition 1.4 (Tree Decomposition)
A tree decomposition *T of a hypergraph G, is a tree where each vertex t is labelled by a set of vertices (also called* bag*) $\lambda(t) \subseteq V(G)$ such that:*

- *$\forall u \in V(G), \exists t \in V(T)$ such that $u \in \lambda(t)$;*
- *$\forall e \in E(G), \exists t \in V(T)$ such that $e \subseteq \lambda(t)$;*
- *$\forall u \in V(G)$, $\forall t_i, t_j \in V(T)$ such that $u \in \lambda(t_i)$ and $u \in \lambda(t_j)$, for all t_k in the only path between t_i and t_j in T, $u \in \lambda(t_k)$.*

Then, from a tree decomposition, one can compute its "width", which is the maximal size of one of its bags (minus one).

Definition 1.5 (Width)
Given a tree decomposition T, the width of T is defined as

$$width(T) = \max_{t \in V(T)} |\lambda(t)| - 1$$

.

Finally, the treewidth of a (hyper)graph is the minimal width of one of its tree decompositions.

Definition 1.6 (Treewidth)
Given a (hyper)graph G, its treewidth is defined as the minimal width among all tree decompositions of G.

This notion is quite useful as it helps to measure the distance between a graph and a tree, for instance, trees have treewidth one, cycles have treewidth two, and if a graph contains a clique of size k then its treewidth is at least $k-1$.

1.1.3 Logical Notions

A logical language $\mathcal{L} = (\mathcal{P}, \mathcal{C})$ is a pair composed of a finite set of predicates $\mathcal{P}$ and a (potentially infinite) set of constants $\mathcal{C}$. Furthermore, we are given an infinite set of variables $\mathcal{V}$. With each predicate of $\mathcal{P}$ is associated a non-negative integer called its *arity*. We do not consider functional symbols except for constants (which are functional symbols of arity 0).

Then, a *term* of $\mathcal{L}$ is either an element of $\mathcal{C}$ (thus a constant), or a variable from $\mathcal{V}$. An *atom* of $\mathcal{L}$ is of the form $p(t_1, \ldots, t_k)$ where p is a predicate from $\mathcal{P}$ of arity k, and $t_1, \ldots, t_k$ are terms from $\mathcal{L}$. It is a *ground atom* if all its terms are constants. For brevity reasons, we sometimes use the expression *p-atom* to denote an atom with predicate p.

Given an atom α, we denote by $terms(\alpha)$ the set of all terms in α, by $var(\alpha)$ the set of all variables in α and by $const(\alpha)$ the set of all constants in α.

In our examples, we denote variables by letters from the end of the alphabet (x,y,z,u,...), and constants by letters from the beginning of the alphabet (a,b,c,...).

Definition 1.7 (Interpretation)
An interpretation *of a logical language $\mathcal{L} = (\mathcal{P}, \mathcal{C})$ is a pair $\mathcal{I} = (D, I)$ where D is a non-empty set called the* interpretation domain *and where I is an* interpretation function *of the symbols of $\mathcal{L}$ such that:*

- *for any $c \in \mathcal{C}$, $I(c) \in D$;*
- *for any $p \in \mathcal{P}$ of arity k, $I(p) \subseteq D^k$.*

An interpretation of $\mathcal{L}$ is a *model* of a formula built on $\mathcal{L}$ if it makes this formula true by considering the classical interpretation of logical connectives and quantifiers.

Definition 1.8 (Logical Consequence, Equivalence)
Given a language $\mathcal{L}$ and two formulae ϕ_1 and ϕ_2 on $\mathcal{L}$, ϕ_2 is a (logical) consequence *of ϕ_1, which is denoted by $\phi_1 \models \phi_2$, if all models of ϕ_1 are models of ϕ_2.*

If $\phi_1 \models \phi_2$ and $\phi_2 \models \phi_1$, ϕ_1 and ϕ_2 are (logically) equivalent*, which is denoted by $\phi_1 \equiv \phi_2$.*

Definition 1.9 (Substitution)
Given a set of variables X and a set of terms T, a substitution *σ of X by T is a mapping from X to T.*

Given an atom α, we denote by $\sigma(\alpha)$ the atom obtained by substituting each occurrence of $x \in var(\alpha) \cap X$ by $\sigma(x)$, i.e., if $\alpha = p(t_1, \ldots, t_k)$ then $\sigma(\alpha) = p(\sigma(t_1), \ldots, \sigma(t_k))$.

Given a set of atoms A, $\sigma(A)$ denotes the set obtained by applying the substitution on each atom, i.e., $\sigma(A) = \{\sigma(\alpha) \mid \alpha \in A\}$.

Definition 1.10 (Homomorphism and Isomorphism)
Given two sets of atoms A_1 and A_2, a homomorphism *from A_1 to A_2 is a substitution π of $var(A_1)$ by $terms(A_2)$ such that $\pi(A_1) \subseteq A_2$. In this case we say that A_1 maps to A_2 by π.*

If π is injective (thus π^{-1} is a function) and π^{-1} is a homomorphism from A_2 to A_1, π is an isomorphism *from A_1 to A_2.*

An isomorphism can also be defined as a bijective substitution σ from $var(A_1)$ to $var(A_2)$ such that $\sigma(A_1) = A_2$. That is why it is often called a "bijective variable renaming".

Furthermore, homomorphisms can also be defined for interpretations.

Definition 1.11 (Homomorphism between Interpretations)
Given two interpretations $\mathcal{I}_1 = (D_1, I_1)$ and $\mathcal{I}_2 = (D_2, I_2)$ of a logical language $\mathcal{L} = (\mathcal{P}, \mathcal{C})$, a homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_2$ is a mapping π from D_1 to D_2 such that:

- *for all $c \in \mathcal{C}$, $\pi(I_1(c)) = I_2(c)$,*
- *for all $p \in \mathcal{P}$ and $(t_1, \ldots, t_k) \in I_1(p)$, $(\pi(t_1), \ldots, \pi(t_k)) \in I_2(p)$.*

The last basic logical notion we need is that of prenex form.

Definition 1.12 (Prenex Form)
A first-order logical formula is in prenex form*, if it is written as a sequence of quantifiers followed by a formula without quantifier, and such that the scope of each quantifier is the whole formula.*

It is well known that every first-order formula can be rewritten into a first-order formula of prenex form. Hence, in the following we always assume that all formulae we are dealing with are under prenex form, which allows us to simplify the various definitions and proofs.

1.1.4 Complexity Classes

In this thesis we make use of several complexity classes. While we do not define here all needed technical notions, we recall the definitions of the classes themselves by increasing complexity.

Definition 1.13 (AC_0)
A problem is in AC_0 if it can be solved by a Boolean circuit of bounded depth with a polynomial number of AND *and* OR *gates.*

Definition 1.14 (NLogSpace (NL))
A problem is in NL if it can be solved by a non-deterministic Turing machine using only a working tape of logarithmic space in the input.

Definition 1.15 (Polynomial Time (PTime))
A problem is in P if it can be solved by a deterministic Turing machine running in polynomial time in the input.

Definition 1.16 (Non-deterministic Polynomial Time (NP))
A problem is in NP if it can be solved by a non-deterministic Turing machine running in polynomial time in the input.

Definition 1.17 (Polynomial Space (PSpace))
A problem is in PSpace if it can be solved by a Turing machine using only a tape of polynomial space in the input. Note that it has been shown that non-deterministic polynomial space is equivalent to deterministic polynomial space.

Definition 1.18 (Exponential Time (ExpTime))
A problem is in ExpTime if it can be solved by a deterministic Turing machine running in simple exponential time in the input.

Furthermore, a problem P is *hard* for a given complexity class C if any instance of a problem from C can be reduced to an instance of P through an "adapted" reduction (in most cases, adapted means "running in polynomial time", but for lower classes (PTime and below), logarithmic space reductions must be used).

Finally a problem P is *complete* for a given complexity class C, if it belongs to C and is hard for C.

For more details about complexity theory notions, the reader is referred to [Pap94].

1.2 Existential Rule Framework

A knowledge base is composed of a set of facts and of an ontology, which is here a set of existential rules. We consider the basic database queries, which are (unions

of) (Boolean) conjunctive queries. In this section we define formally these objects as well as the associated notions that allow to do reasoning. The main problem we study throughout this manuscript is the conjunctive query entailment problem, which asks if a (Boolean) conjunctive query is a logical consequence of a knowledge base.

In most settings, a fact is a ground atom. However, it is convenient to consider facts with existentially quantified variables, which naturally leads to see a fact as an existentially closed conjunction of atoms. Furthermore, the prenex form of a conjunction of facts is itself a fact, hence we can identify the notions of a fact and a set of facts.

Definition 1.19 (Set of Facts)
Given a logical language $\mathcal{L}$, a fact *or* set of facts *is an existentially closed conjunction of atoms on $\mathcal{L}$.*

In the following we will often consider sets of facts as sets of atoms, which allows to use set theoretic notions such as the inclusion on sets of facts. We also often omit the existential quantifiers in the representation of facts, since there can be no ambiguity.

Example 1.1 (Set of Facts)
Consider the first-order formula $\mathcal{F}$: $\exists x \exists y (p(x, y) \wedge q(y, a, x) \wedge s(x))$; where a is a constant.

The formula $\mathcal{F}$ is an existentially closed conjunction of atoms, therefore a set of facts, which we can also denote by $\{p(x, y), q(y, a, x), s(x)\}$.

Sets of facts can also be seen as (hyper)graphs, which allows to apply several graph notions to them. Indeed, given a set of facts $\mathcal{F}$, one can build a directed hypergraph whose set of vertices is in bijection with the set of terms of $\mathcal{F}$, and whose set of hyperedges is in bijection with the set of atoms of $\mathcal{F}$. Then each vertex is labelled by the corresponding term, and each edge by the predicate of the corresponding atom.

Example 1.2 (Graphical View of a Set of Facts)
Consider the set of facts $\mathcal{F}$ from Example 1.1. Figure 1.1 depicts its graphical representation, where hyperedges are represented via circle nodes.

Definition 1.20 (Conjunctive Query, Union of Conjunctive Queries)
A conjunctive query *(CQ) is a conjunction of atoms where all variables are either free (called* answer variables*) or existentially quantified.*

A union of conjunctive queries *(UCQ) is a disjunction of CQs with the same answer variables.*

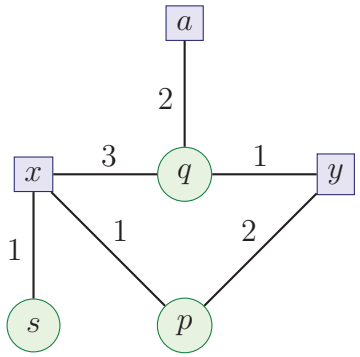

Figure 1.1: Graphical representation of the set of facts $\mathcal{F}$ from Example 1.1

If all variables of a conjunctive query Q are quantified (thus Q has the exact same form as a set of facts), we say that Q is a *Boolean conjunctive query* (BCQ). Throughout this thesis we only consider Boolean conjunctive queries that we simply call conjunctive queries.

A well-known fundamental result is that the logical consequence on existentially closed conjunctions of atoms (e.g., two sets of facts or Boolean conjunctive queries) amounts to the existence of a homomorphism as stated by the next theorem, which has been proven in several contexts (see, e.g., [AHV95] for a version in terms of classical CQ containment).

Theorem 1.1 (Folklore)

Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two sets of facts. There exists a homomorphism from $\mathcal{F}_1$ to $\mathcal{F}_2$ if and only if $\mathcal{F}_2 \models \mathcal{F}_1$.

Since one can compare two sets of facts with respect to logical consequence, it is natural to consider the core of a set of facts, which is a minimal equivalent subset. It should be pointed out that the definition of core we give here is restricted to finite sets of facts, which is enough for our needs. However, a more technical definition for infinite sets exists [Bod05].

Definition 1.21 (Core)
Given a finite set of facts $\mathcal{F}$, a core *of $\mathcal{F}$ is a minimal subset of $\mathcal{F}$ equivalent to $\mathcal{F}$.*

Example 1.3 (Core)
Consider the set of facts $\mathcal{F} = p(x, y) \wedge p(y, z) \wedge p(x, u) \wedge p(u, z)$. A core of $\mathcal{F}$ is $p(x, y) \wedge p(y, z)$.

It is well-known that if $\mathcal{F}$ is a finite set of facts, then all its cores are isomorphic, for instance in Example 1.3, the set of facts $p(x,u) \wedge p(u,z)$ is also a core of $\mathcal{F}$.

Furthermore, given a set of facts $\mathcal{F}$, one can define its "isomorphic model", which has the same structure as $\mathcal{F}$.

Definition 1.22 (Isomorphic Model)
Given a set of facts $\mathcal{F}$ built on the logical language $\mathcal{L} = (\mathcal{P}, \mathcal{C})$, the isomorphic model *of $\mathcal{F}$ denoted by $M(\mathcal{F}) = (D, I)$ is defined as follows:*

- *D is in bijection with $terms(\mathcal{F}) \cup \mathcal{C}$ (to simplify notations we consider that this bijection is the identity),*
- *for all $c \in \mathcal{C}$, $I(c) = c$,*
- *for all $p \in \mathcal{P}$, $I(p) = \{(t_1, \ldots, t_k) \mid p(t_1, \ldots, t_k) \in \mathcal{F}\}$.*

Finally, it is also well-known that the following properties are equivalent: given two sets of facts $\mathcal{F}_1$ and $\mathcal{F}_2$,

1. $\mathcal{F}_1 \models \mathcal{F}_2$,
2. there is a homomorphism from $\mathcal{F}_2$ to $\mathcal{F}_1$,
3. $M(\mathcal{F}_1)$ is a model of $\mathcal{F}_2$.

An existential rule is a positive (and conjunctive) rule. Its special feature is that its head may introduce variables that do not occur in its body, and which are existentially quantified, hence the name "existential rule". This allows to infer the existence of unknown individuals (possibly equal to (known) individuals appearing in the set of facts).

Definition 1.23 (Existential Rule)
An existential rule *(or simply* rule*) on a logical language $\mathcal{L}$ is a closed formula of form $\forall x_1, \ldots, \forall x_b (B \rightarrow \exists z_1, \ldots, z_h H)$, where B and H are two finite conjunctions of atoms on $\mathcal{L}$, $var(B) = \{x_1, \ldots, x_b\}$ and $var(H) \setminus var(B) = \{z_1, \ldots, z_k\}$.*

The sets of atoms B and H are respectively called the body *and the* head *of R. Variables occurring in $var(B) \cap var(H)$ are called the* frontier variables *of R and denoted by $fr(R)$. Variables occurring in $var(H) \setminus var(B)$ are called* existential variables.

Since no ambiguity can arise, we omit the quantifiers in rules.

Example 1.4 (Existential Rule)
Consider the rule $R = \forall x \forall y (p(x,y) \rightarrow \exists z (p(y,z)))$. This rule will also be denoted by $R = p(x,y) \rightarrow p(y,z)$. Variable y is the only frontier variable of R and z is its only existential variable.

Two other kinds of rules may be added to the framework: *negative constraints*, which are rules with an absurd head ($B \rightarrow \bot$), and *equality rules* which are rules whose head specifies the equality between two terms ($B \rightarrow t_1 = t_2$). For details, see [BLMS11, CGL09]. It should be pointed out that there is no additional complexity (or decidability) cost for considering negative constraints (except if only particular BCQ are considered). By contrast, equality rules are known for being hard to handle from both complexity and decidability points of view. In the following, we restrict our focus to knowledge bases without negative constraints nor equality rules. We are now ready to formally define a knowledge base.

Definition 1.24 (Knowledge Base)
A knowledge base *is a pair* $\mathcal{K} = (\mathcal{F}, \mathcal{R})$ *where* $\mathcal{F}$ *is a (finite) set of facts and* $\mathcal{R}$ *is a (finite) set of existential rules.*

Note that, without loss of generality, we will always assume that sets of facts and all rules use disjoint sets of variables.

In the following, when we consider logical notions involving knowledge bases, sets of rules and/or facts, we implicitly consider the associated formula which is the conjunction of all involved formulae. In particular, $(\mathcal{F}, \mathcal{R})$ is seen as the conjunction of the formula associated with $\mathcal{F}$ and of all the rules from $\mathcal{R}$.

A nice property of this setting is that given a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R})$, there always exists a model of $\mathcal{K}$ that can be considered as a representative of all models of $\mathcal{K}$, in the sense that is is sufficient to consider this model to check entailment from $\mathcal{K}$. Such a model has the property of being "universal".

Definition 1.25 (Universal Model)
Given a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R})$*, a* universal model M *of* $\mathcal{K}$ *is a model of* $\mathcal{K}$ *such that for all models* M' *of* $\mathcal{K}$*, there is a homomorphism from* M *to* M'*.*

Note that not all knowledge bases have a finite universal model.

Now that we have defined all of our basics objects, we can reformulate the entailment problem more formally:

Problem 1.1 (Conjunctive Query Entailment)

Given a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R})$ and a Boolean conjunctive query Q, the *conjunctive query entailment* problem (CQ entailment) asks the following question: does $\mathcal{K} \models Q$ hold?

It is well-known that this problem is undecidable (e.g., [BV81]), even under strong restrictions such as using a single rule or restricting the vocabulary (a single binary predicate, no constant, ...), see, for instance, [BLMS11].

However many restrictions on the set of rules are known to ensure decidability. Most of them can be classified in three families, which are associated with forward or backward chaining schemes [BLMS11]. The first one is that of Finite Expansion Sets (FES). It ensures that a finite universal model of the knowledge base exists. This property allows for some finite Forward Chaining, a process that "applies" rules until an answer to the query is found or nothing "useful" is produced. The second family is called Finite Unification Sets (FUS) and guarantees that some Backward Chaining method halts. This process uses the rules to "rewrite" the query which creates a UCQ, which is then checked against the initial set of facts. Finally, the last family is called Bounded Treewidth Set (BTS) [CGK08]. It ensures that the potentially infinite universal model of the knowledge base has a bounded treewidth. This class does not give an algorithm yet, but thanks to Courcelle's theorem [Cou89], we know that entailment over BTS knowledge bases is decidable. Furthermore, an expressive subclass of BTS, namely Greedy BTS (GBTS) is provided with an algorithm [BMRT11, Tho13].

These classes will be precisely defined in Chapter 2.

1.2.1 Forward Chaining

Rules allow to infer new knowledge from an initial set of facts, giving rise to the notion of rule application. This notion is defined as usual in rule languages, i.e., the head of the rule is added to the set of facts according to a homomorphism from the body of the rule to the set of facts. The only specificity for existential rules is that existential variables in the rule head are replaced by fresh variables.

Definition 1.26 (Rule Applicability)
A rule $R = B \to H$ is applicable *to a set of facts $\mathcal{F}$ if there exists a homomorphism π from B to $\mathcal{F}$.*

Definition 1.27 (Rule Application)
Given a set of facts $\mathcal{F}$, a rule $R = B \to H$ and a homomorphism π from B to $\mathcal{F}$, the result of the application of the rule *R on $\mathcal{F}$ according to π is defined as*

$$\alpha(\mathcal{F}, R, \pi) = \mathcal{F} \cup \pi^{safe}(H)$$

where π^{safe} if the extension of π to the existential variables of R, i.e.: for all variables x occurring in H, $\pi^{safe}(x) = \pi(x)$ if $x \in fr(R)$ and $\pi^{safe}(x) = z$ otherwise, where z is a fresh variable (i.e., a variable that does not occur elsewhere).

Intuitively, π^{safe} is a "safe" extension of π that maps all existential variables of H to some distinct fresh variables, in order to avoid confusion with variables already occurring in $\mathcal{F}$.

Example 1.5 (Rule Application)
Let $\mathcal{F} = \{c(a)\}$ be a set of facts and $R = c(x) \rightarrow p(x, y) \wedge c(y)$ be an existential rule.

Rule R is applicable to $\mathcal{F}$ since there exists the following homomorphism $\pi = \{x \mapsto a\}$, and the application of R on $\mathcal{F}$ according to π produces the set of facts $F_1 = \alpha(\mathcal{F}, R, \pi) = c(a) \wedge p(a, y_1) \wedge c(y_1)$.

Then rule R is applicable to F_1 thanks to the homomorphism $\pi_2 = \{x \mapsto y_1\}$, which would produce $p(y_1, y_2) \wedge c(y_2)$ (and we could go on like this since each added c-atom allows for a new application of R).

A sequence of rule applications is called a derivation and is defined as follows.

Definition 1.28 (Derivation)
Given a set of facts $\mathcal{F}$ and a set of rules $\mathcal{R}$, a derivation *of $\mathcal{F}$ w.r.t. $\mathcal{R}$ (also called $\mathcal{R}$-derivation) is a (potentially infinite) sequence D of triples of the form (R_i, π_i, F_i) such that $D_0 = (\emptyset, \emptyset, F_0 = \mathcal{F})$, and $\forall i > 0$ in $dom(D), R_i = (B_i, H_i) \in \mathcal{R}, \pi_i$ is a homomorphism from B_i to F_{i-1} and $F_i = \alpha(F_{i-1}, R_i, \pi_i)$.*

When D is of length $k \in \mathbb{N}$, we say that it is a derivation from $\mathcal{F}$ to F_k.

Example 1.6 (Derivation)
Consider again the set of facts $\mathcal{F} = c(a)$, and the set of rules $\mathcal{R}$ composed of a single rule $R = c(x) \rightarrow p(x, y), c(y)$ from Example 1.5.

A derivation D of $\mathcal{F}$ with respect to $\mathcal{R}$ could be:

$\mathbf{i}$	$\mathbf{R_i}$	$\pi_\mathbf{i}$	$\mathbf{F_i}$
0	$\emptyset$	$\emptyset$	$\mathcal{F}$
1	R	$\{x \mapsto a\}$	$\mathcal{F} \cup \{p(a, y_1), c(y_1)\})$
...	...	...	...
k	R	$\{x \mapsto y_{k-1}\}$	$F_{k-1} \cup \{p(y_{k-1}, y_k), c(y_k)\}$
...	...	...	...

Note that in this example, only a single derivation (not using the same homomorphism twice) is possible, but in general, this is not the case.

The next theorem states that derivations are a sound and complete way to "solve" the CQ entailment problem.

Theorem 1.2 ([BLMS11])

Let $\mathcal{F}$ be a set of facts, $\mathcal{R}$ be a set of rules, and Q be a Boolean conjunctive query; then, $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if there exists a (finite) $\mathcal{R}$-derivation from $\mathcal{F}$ to F_k such that $F_k \models Q$.

Chapter 3 is devoted to the family of forward chaining algorithms and to rule classes for which some of these algorithms always halt.

1.2.2 Backward Chaining

In contrast with forward chaining, backward chaining mecanisms start from the query. In Prolog and other logic programming languages, facts are ground atoms (which possibly involve functional symbols) and are seen as rules with an empty body. The aim is then to "erase" the query by iteratively rewriting it with the rules. In knowledge representation, we clearly distinguish between rules and facts, hence we decompose the backward chaining into two processes: rewriting the query using the rules, and mapping a rewritten query to the set of facts (by homomorphism). The OBDA approach goes one step further by completing the rewriting process before trying to map the obtained query (or set of queries) to the set of facts. The aim is to come back to a classical database query answering problem.

Differently from classical unifiers used in logic programming which unify an atom from the query with the head atom, piece-unifiers do not consider a single atom of the query but a subset of the query. This allows to correctly process existential variables as illustrated by the next example.

Example 1.7 (Classical Unifier)
Consider the rule $R = person(x) \rightarrow hasParent(x, z)$ *which says that every person has a parent, and the set of facts* $\mathcal{F} = person(Bob) \wedge painter(Max)$. *Assume one asks if someone has a parent who is a painter with the following query* $Q = \exists u, v(hasParent(u, v) \wedge painter(v))$. *With a classical (most general) unifier of* Q *with the head of* R, *for instance* $\{x \mapsto u, z \mapsto v\}$, *we obtain the direct rewriting* $Q_1 = \exists u, v(person(u) \wedge painter(v))$ *(in which the connection between* u *and* v *has been lost). We then have that* $\mathcal{F} \models Q_1$ *while* $(\mathcal{F}, \mathcal{R}) \not\models Q$. *Hence, this rewriting is unsound. Here the "piece condition" from the following definition of a piece-unifier has been violated.*

Piece-unifiers can be seen as generalised unifiers, in the sense that they unify two sets of atoms. Moreoever they process existential variables in a special way.

The following definition of a piece-unifier considers the partition on terms induced by a (generalised) unifier: the terms unified together are in the same class of the partition. Partitions have nice properties that are convenient in proofs.

Definition 1.29 (Piece-unifier)
Given a rule $R = (B, H)$ *and a set of atoms* Q, *a* piece-unifier *of* Q *with* H *is a triple* $\mu = (Q', H', P_\mu)$ *where* $Q' \subseteq Q$, $H' \subseteq H$ *and* P_μ *is a partition of terms of* $Q' \cup H'$ *such that:*

(i) **Admissibility:** *there is at most one single constant or existential variable from* H' *in each class of* P_μ, *and if a class contains an existential variable, it cannot contain a frontier variable from* R *(hence, the other terms in the class are from* Q'*);*

(ii) **Unifiability:** $\sigma_\mu(Q') = \sigma_\mu(H')$, *where* σ_μ *is a substitution associated with the partition* P_μ *(formally defined below);*

(iii) **Piece condition:** *if there is an existential variable z occurring in H' then no atom from $Q \setminus Q'$ may contain a variable x occurring in the same class as z.*

Definition 1.30 (Substitution Associated with a Partition)
Given an admissible partition of a set of terms P_μ, a substitution associated with P_μ is obtained as follows: for each class $C \in P_\mu$, choose a term $t_C \in C$ (if C contains a constant, then t_C must be that constant); then $\sigma_\mu = \bigcup_{C \in P} \{t \mapsto t_C \mid \forall t \in C\}$. The terms t_C are said to be preserved *by the substitution.*

Note that when a class contains only variables, one can choose to give priority to variables occurring in the query or in the rule.

Example 1.8 (Piece-unifier)
Consider the rule $R = p(x, y) \rightarrow s(x, z)$ and the conjunctive query $Q = s(a, u) \wedge s(v, u) \wedge r(v)$, where a is a constant. There is the following piece-unifier of Q with H:

$$\{Q' = \{s(a, u), s(v, u)\}, H' = \{s(x, z)\}, P_\mu = \{\{a, v, x\}, \{u, z\}\}\}$$

Indeed, no class of P_μ contains more than one existential variable from H' or constant, and the class that contains an existential variable does not contain any frontier variable, thus admissibility is satisfied. One can choose for instance the substitution $\sigma_\mu = \{v \mapsto a, x \mapsto a, u \mapsto z\}$. We then check that $\sigma_\mu(Q') = \sigma_\mu(H')$. Finally all atoms of Q using variable u (which is unified with the existential variable z) belong to Q', hence the piece condition is satisfied.

Note that no smaller piece-unifier of Q with H exists since u is necessarily unified with z, hence the piece condition requires that both atoms of Q are part of Q'.

Given this extended notion of unifier, we can now define a direct query rewriting as follows.

Definition 1.31 (Direct Query Rewriting)
Given a rule $R = (B, H)$, a conjunctive query Q, a piece-unifier $\mu = (Q', H', P_\mu)$ of Q with H and a substitution σ_μ associated with P_μ, the direct query rewriting *of Q according to μ is the following CQ: $\beta(Q, R, \mu) = \sigma_\mu(Q \setminus Q') \cup \sigma_\mu(B)$.*

Note that in the above definition any substitution associated with μ can be used, as they all give isomorphic results.

Example 1.9 (Direct Rewriting)
Consider the rule R, the query Q and the piece-unifier μ from Example 1.8. The direct rewriting of Q according to μ is: $\beta(Q, R, \mu) = r(a) \wedge p(a, y)$.

Then a rewriting sequence is defined as a sequence of direct rewriting steps.

Definition 1.32 (Rewriting Sequence)
Given a CQ Q, and a set of rules $\mathcal{R}$, a rewriting sequence *of Q w.r.t. $\mathcal{R}$ (also called an $\mathcal{R}$-rewriting sequence) is a (potentially infinite) sequence S of triples (R_i, μ_i, Q_i) such that $S_0 = (\emptyset, \emptyset, Q)$ and $\forall i > 0, R_i = (B_i, H_i) \in \mathcal{R}$, μ_i is a piece-unifier of Q_{i-1} with H_i and $Q_i = \beta(Q_{i-1}, R_i, \mu_i)$.*

When S is of length $k \in \mathbb{N}$, we say that it is a rewriting sequence from Q to Q_k.

Similarily to derivations, rewriting sequences are a sound and complete way to "solve" the CQ entailment problem as stated by the next theorem.

Theorem 1.3 ([BLMS11])

Let $\mathcal{F}$ be a set of facts, $\mathcal{R}$ be a set of rules and Q be a Boolean conjunctive query; then, $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if there exists a rewriting sequence from Q to Q_k such that $\mathcal{F} \models Q_k$.

This notion gives naturally rise to an algorithm that reads: let $\mathcal{Q} = \{Q\}$, for all queries $Q_i \in \mathcal{Q}$, rewrite Q_i according to some unifier, then add the result to $\mathcal{Q}$ if it is not equivalent to another query. If at some point no new query is added, end the process and evaluate the union of conjunctive queries on the knowledge base.

While both of these scheme families seem quite different, they are actually strongly related in the sense that given a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R})$ and a Boolean conjunctive query Q, if there exists a derivation of $\mathcal{F}$ w.r.t. $\mathcal{R}$ of length k such that F_k entails Q, then there exists a rewriting sequence of at most length k leading to a query Q' entailed by $\mathcal{F}$; and if there exists a rewriting sequence of length k such that $\mathcal{F}$ entails Q_k, then there exists a derivation of same length leading to a set of facts F_k such that F_k entails Q, as stated by the next theorem.

Note that the derivation may be longer than the rewriting sequence, since in the former one can add rule applications that are useless to answer Q.

Theorem 1.4 ([BLMS11])

Let $\mathcal{F}$ be a set of facts, $\mathcal{R}$ be a set of rules and Q be a conjunctive query. If there exists a derivation from $\mathcal{F}$ to F_k such that $F_k \models Q$, then there exists $k' \leq k$ and a rewriting sequence from Q to $Q_{k'}$ such that $\mathcal{F} \models Q_{k'}$.
Furthermore, if there exists a rewriting sequence from Q to Q_k such that $\mathcal{F} \models Q_k$, then there exists a derivation from $\mathcal{F}$ to F_k such that $F_k \models Q$.

1.3 Some Useful Translations

In this section we define several transformations of sets of rules (with one of them also involving a translation of the set of facts), mostly used to simplify the shape of the rules we need to consider.

The first translation is used to convert rules into single-piece headed rules.

We define first the "piece graph" of a rule that allows us to easily define the notion of piece.

Definition 1.33 (Piece Graph)
Given a rule $R = (B, H)$ its piece graph *is defined as the undirected graph whose set of vertices is the set of atoms occurring in H, and where there is an edge between two vertices corresponding to atoms h_1 and h_2 in H if there is an existential variable z such that $z \in var(h_1)$ and $z \in var(h_2)$.*

Definition 1.34 (Rule Piece, Single-piece Headed Rule)
Given a rule $R = (B, H)$, a rule piece *of R is the set of atoms corresponding to a connected component of the piece graph of R.*

A rule R is a single-piece headed rule *if H is a single rule piece of R.*

Intuitively, a rule piece corresponds to the minimal unit of information contained in a rule head. Indeed, any rule can be decomposed into an equivalent set of rules with the same body and a head restricted to a single rule piece as performed by the following translation.

Definition 1.35 (Single-piece Translation)
Given a set of rules $\mathcal{R}$, its single-piece translation *$sp(\mathcal{R})$ is defined as follows. For each rule $R = (B, H) \in \mathcal{R}$, for each rule piece H_i of R, add in $sp(\mathcal{R})$ the rule $B \to H_i$.*

This transformation is without loss of generality, hence, we can always assume that all rules we consider are single-piece.

Proposition 1.1

Given a set of rules $\mathcal{R}$, $\mathcal{R}$ and $sp(\mathcal{R})$ are logically equivalent.

Example 1.10 (Single-piece Translation)
Consider the following rule (where c is a constant and every other term a variable): $R = p(x, y, c) \to q(y, z_1) \wedge r(z_1) \wedge s(x, c, z_2)$.

The piece graph of R is depicted on Figure 1.2.

The single-piece translation of $\{R\}$ is the following set of rules $\mathcal{R}'$:

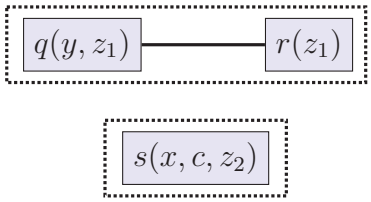

Figure 1.2: Piece Graph of rule R from Example 1.10

- $R_1 = p(x, y, c) \to q(y, z_1) \wedge r(z_1)$
- $R_2 = p(x, y, c) \to s(x, c, z_2)$

The next translation further simplifies the shape of rules we consider.

Definition 1.36 (Atomic-headed Translation)
Given a set of rules $\mathcal{R}$ its atomic-headed translation *$ah(\mathcal{R})$ is defined as follows. For each rule $R = (B, H) \in sp(\mathcal{R})$ where $|H| > 1$, perform the following:*

1. *create a new predicate r_R of arity $|var(H)|$,*
2. *add to $ah(\mathcal{R})$ the rule $B \to r_R(\vec{t})$ where $\vec{t}$ is the set of variables occurring in H,*
3. *for each atom $h_i \in H$, add the rule $r_R(\vec{t}) \to h_i$ to $ah(\mathcal{R})$.*

This translation creates new predicates with arity depending on the size of a rule head, and thus is not suitable in a context where we consider that the arity is bounded. However, if the arity is not assumed to be bounded, it will prove useful to be able to simplify rule heads into single atoms.

Example 1.11 (Atomic-headed Translation)
Consider again the rule R from Example 1.10, and its single-piece translation $sp(R) = \{R_1, R_2\}$. Its atomic-headed translation is the following set of rules:

- $R_{1,a} = p(x, y, c) \to r_{R_1}(y, z_1)$
- $R_{1,b_1} = r_{R_1}(y, z_1) \to q(y, z_1)$
- $R_{1,b_2} = r_{R_1}(y, z_1) \to r(z_1)$
- $R_2 = p(x, y, c) \to s(x, c, z_2)$

Note that this translation could be defined on any set of rules $\mathcal{R}$ without first decomposing it into $sp(\mathcal{R})$, however it would potentially create predicates with an arity larger than needed.

Proposition 1.2

Let $\mathcal{R}$ be a set of rules. For any CQ Q and knowledge base $(\mathcal{F}, \mathcal{R})$ on a logical language $\mathcal{L}$, $(\mathcal{F}, \mathcal{R}) \models Q$ iff $(\mathcal{F}, ah(\mathcal{R})) \models Q$.

The last translation we consider is used to get rid of constants occurring in rules. As for the atomic-headed translation, this cannot be used without loss of generality in the case where the arity is bounded. Furthermore, this translation not only modifies the rules, but also the set of facts. The idea is to totally order the constants, say from c_1 to c_n, and increase the arity of each predicate by creating a placeholder for each constant. Then rules are translated in such a way that $p(\dots, c, \dots)$ where c is the i^{th} constant becomes $p'(\dots, x_i, \dots, x_1, \dots, x_n)$, where x_i stands for the i^{th} constant. Hence the occurrence of the i^{th} constant in an atom is replaced by the double occurrence of the variable x_i. Facts are translated by adding $c_1, \dots, c_n$ at the end of each atom.

Definition 1.37 (No-constant Translation)
Given a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R})$, its no-constant translation *$nc(\mathcal{K})$ is defined as follows. For each predicate p of arity k, create a new predicate p' with arity $k + |cst(\mathcal{K})|$, where $cst(\mathcal{K}) = \{c_1, \dots, c_n\}$ is the set of all constants occurring either in $\mathcal{F}$ or in some rule $R \in \mathcal{R}$.*

Then for each atom $\alpha = p(t_1, \dots, t_k)$ in some rule $R \in \mathcal{R}$, replace α by an atom $p'(t'_1, \dots, t'_k, \dots, t'_{k+n})$ where (all x_j being distinct fresh variables):

$$t'_i = \begin{cases} t_i & \text{if } i \leq k \text{ and } t_i \text{ is a variable,} \\ x_j & \text{if } i \leq k \text{ and } t_i = c_j, \\ x_j & \text{if } i = k + j. \end{cases}$$

Now for each atom $\alpha = p(t_1, \dots, t_k)$ in $\mathcal{F}$, replace α by an atom $p'(t_1, \dots, t_k, c_1, \dots, c_n)$. Finally, for each predicate p create the rule $p'(y_1, \dots, y_k, x_1, \dots, x_n) \to p(y_1, \dots, y_k)$, where all variables are distinct.

Example 1.12 (No-constant Translation)
Consider again rule $R = p(x, y, c) \to q(y, z_1) \wedge r(z_1) \wedge s(x, c, z_2)$ from Example 1.10, and let $\mathcal{F} = p(b, b, b) \wedge p(b, b, c)$ be a set of facts, where b is a constant.

The no-constant translation of $\mathcal{K} = (\mathcal{F}, \{R\})$ is obtained as follows. The set of constants is $\{b, c\}$, hence $n = 2$. First we create four new predicates p', q', r' and s' of arity 5, 4, 3 and 5 respectively. Now we update rule R into:

$$R' = p'(x, y, x_2, x_1, x_2) \to q'(y, z_1, x_1, x_2) \wedge r'(z_1, x_1, x_2) \wedge s'(x, x_2, z_2, x_1, x_2)$$

Then the set of facts becomes $\mathcal{F}' = p(b, b, b, b, c) \wedge p(b, b, c, b, c)$. Finally we add the following four rules (one for each predicate):

- $R_p = p'(y_1, y_2, y_3, x_1, x_2) \to p(y_1, y_2, y_3)$,
- $R_q = q'(y_1, y_2, x_1, x_2) \to q(y_1, y_2)$,
- $R_r = r'(y_1, x_1, x_2) \to r(y_1)$,
- $R_s = s'(y_1, y_2, y_3, x_1, x_2) \to s(y_1, y_2, y_3)$.

These four rules allow to transfer facts from the new vocabulary to the original one.

Proposition 1.3

Let $\mathcal{R}$ be a set of rules. For any CQ Q and knowledge base $(\mathcal{F}, \mathcal{R})$ on a logical language $\mathcal{L}$, $(\mathcal{F}, \mathcal{R}) \models Q$ iff $nc(\mathcal{F}, \mathcal{R}) \models Q$.

In particular, in Section 4.2, we will make the assumption that all rules are single-headed rules and contain no constant, and this will allow for simpler decidability proofs.

Furthermore, note that both the atomic-headed and the no-constant translations only preserve answers on the original vocabulary. Indeed, new predicates are created, and thus, queries that would involve these new predicates may have a positive answer in the translation but not in the original knowledge base.

Conclusion

The next chapters will rely on the fundamental notions we have introduced here:

- **Chapter 2:** this chapter is devoted to rule classes for which the CQ entailment problem is decidable. Derivations and rewriting sequences are the fundamental mechanisms that underly these classes.
- **Chapter 3:** several forward chaining (or "chase") mechanisms can be based on the derivation notion, and after recalling most of them, we study the various acyclicity conditions that ensure the finiteness of some forward chaining variant.
- **Chapter 4:** we study the compatibility of transitivity rules with other known decidable classes of rules. The algorithm we develop for the case of linear rules is based on backward chaining techniques.

Chapter 2

Landscape of Decidable Classes of Rules

Contents

In this chapter we review various decidability and complexity results from the literature for the CQ entailment problem. More specifically, we recall the main classes of rules C such that entailment over knowledge bases where the set of rules belongs to C is decidable. For brevity reasons, we often refer to them as "decidable classes of rules". We end this chapter by pointing out the relationships between description logics and the existential rule framework.

2.1 Abstract Rule Classes

First, we recall three abstract rule classes, "abstract" in the sense that we cannot decide in general whether a set of rules belongs to one of them [BLMS11]. These classes ensure a specific behaviour of either some forward or some backward chaining algorithm.

Definition 2.1 (Finite Expansion Set (FES) [BLMS11])
Given a set of rules $\mathcal{R}$, we say that $\mathcal{R}$ satisfies the Finite Expansion Set *(FES) property if for any set of facts $\mathcal{F}$, there exists some $k \in \mathbb{N}$ such that for any query Q, there exists a (finite) derivation from $\mathcal{F}$ to F_k such that $F_k \models Q$ if and only if $(\mathcal{F}, \mathcal{R}) \models Q$.*

The Finite Expansion Set property ensures that for any set of facts $\mathcal{F}$, the knowledge base $(\mathcal{F}, \mathcal{R})$ has a finite universal model. Hence a forward chaining algorithm that applies rules in a "breadth-first" manner (also known as the "chase"), and after each rule application (or after each breadth-first step) considers the core of the obtained set of facts halts on any knowledge base built with a set of rules satisfying the FES property. Then, the isomorphic model of the saturated set is finite. More details on the different kinds of chase and their properties are given in Section 3.1.

Example 2.1 (Non-FES)
The set of rules composed of the single rule $R = q(x) \rightarrow p(x, y) \wedge q(y)$ *is not FES. Indeed, consider the set of facts* $q(a)$*. Then one can apply rule* R *according to homomorphism* $\pi_1 = \{x \mapsto a\}$ *and obtain* $F_1 = q(a) \wedge p(a, y_0) \wedge q(y_0)$*. A core of* F_1 *if* F_1 *itself. The rule* R *can be applied again, this time using the homomorphism* $\pi_2 = \{x \mapsto y_0\}$ *and the resulting set of facts is* $F_2 = q(a) \wedge p(a, y_0) \wedge q(y_0) \wedge p(y_0, y_1) \wedge q(y_1)$*. One can see that this rule can be applied again, and this process will never halt.*

The second abstract property relies on query rewriting. A set of rewritings $\mathcal{Q}$ of a query Q with a set of rules $\mathcal{R}$ is said to be *sound* and *complete* if for any set of facts $\mathcal{F}$, it holds that $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if there is $Q_i \in \mathcal{Q}$ such that $\mathcal{F} \models Q_i$ (the direction $\Rightarrow$ expresses the soundness while the direction $\Leftarrow$ the completeness). When $\mathcal{Q}$ is finite, it can be seen as a union of conjunctive queries. Then the previous condition can be recast as: $(\mathcal{F}, \mathcal{R}) \models Q$ iff $\mathcal{F} \models \mathcal{Q}$.

Definition 2.2 (Finite Unification Set [BLMS11])
Given a set of rules $\mathcal{R}$*, we say that* $\mathcal{R}$ *satisfies the* Finite Unification Set *(FUS) property if for any query* Q*, there is a finite sound and complete set of rewritings of* Q *with* $\mathcal{R}$*.*

This set is clearly computed independently from any set of facts. Note that the FUS property can also be expressed as follows: a set of rules $\mathcal{R}$ is FUS if and only if for any conjunctive query Q, there is some $k \in \mathbb{N}$ such that the set of all (sound) rewritings of Q with $\mathcal{R}$ that can be obtained by a rewriting sequence of length less than or equal to k is complete. In other words, Q is "UCQ-rewritable".

This property ensures that given a set of rules $\mathcal{R}$ that satisfies it, any conjunctive query Q can be rewritten into a (finite) union of conjunctive queries $\mathcal{Q}$ such that for any set of facts $\mathcal{F}$, $(\mathcal{F}, \mathcal{R})$ entails Q if and only if there is a rewriting $Q' \in \mathcal{Q}$, such that $\mathcal{F}$ entails Q'. Note that a FUS set of rules may have an infinite sound set of rewritings. However this set always has a finite complete subset obtained by ignoring redundant queries, i.e., queries Q_2 that are contained in another query Q_1 (which can be checked by exhibiting a homomorphism from Q_1 to Q_2). In other words, for any query Q there exists some $k \in \mathbb{N}$ such that the set $\mathcal{Q}$ of all sound rewritings of Q obtained by a sequence of length less than or equal to k is complete. Hence, if a knowledge-base is built on a finite unification set of rules, then a query

rewriting algorithm that would after each rewriting only keeps the queries that are not contained in another one halts, see [KLMT15] for such an algorithm.

It has been shown in [LMU16] that FUS is equivalent to the *Bounded Derivation Depth Property* (BDDP) [CGL09] defined as follows: for any query Q, there exists some $k \in \mathbb{N}$ such that for any set of facts $\mathcal{F}$, $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if there exists a (finite) derivation from $\mathcal{F}$ to F_k such that $F_k \models Q$. This allows to relate the notions of derivations and rewriting sequences for FUS rules: for any query Q there exists some $k \in \mathbb{N}$ such that for any set of facts $\mathcal{F}$, $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if there is a (finite) derivation from $\mathcal{F}$ to F_k such that $F_k \models Q$ if and only if there is a (finite) rewriting sequence from Q to Q_k such that $\mathcal{F} \models Q_k$.

Furthermore, in [GKK+14], it is shown that the BDDP is equivalent to First-order Rewritability, i.e., to the existence of a first-order formula $\varphi(Q)$ such that $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if $\mathcal{F} \models \varphi(Q)$. While a UCQ rewriting is already a specific kind of first-order rewriting, there may exist a first-order rewriting exponentially smaller than all UCQ rewritings. In turn, every first-order rewriting can be translated into an equivalent UCQ-rewriting (which, however, can be exponentially larger). Such rewriting can thus be useful from a practical point of view, since most databases query languages (such as SQL) allows for first-order querying.

Example 2.2 (Non-FUS)
The following set of rules $\{p(x, y) \wedge p(y, z) \rightarrow p(x, z)\}$ *is not FUS. For instance the query* $Q = p(a, b)$ *does not admit a finite sound and complete set of rewritings. Indeed, any sound and complete set of rewritings of* Q *contains* $p(a, b)$ *and the queries* $p(a, y_0) \wedge p(y_0, y_1) \wedge \ldots p(y_k, b)$ *for all* $k \in \mathbb{N}$*. Note that all these queries are pairwise incomparable with respect to containment.*

The last abstract property relies on the notion of treewidth of a set of facts. To compute such treewidth, we use the *primal graph* of a set of facts $\mathcal{F}$, which is the undirected graph defined as follows: the set of vertices is the set of terms occurring in $\mathcal{F}$ and there is an edge between two vertices u and v if there is an atom $\alpha \in \mathcal{F}$ such that the terms corresponding to u and v occur both in α. Then the treewidth of $\mathcal{F}$ is the treewidth of the primal graph of $\mathcal{F}$.

Definition 2.3 (Bounded Treewidth Set (BTS) [CGK08, BLMS11])
A set of rules $\mathcal{R}$ *is a* bounded treewidth set *(BTS) if for any set of facts* $\mathcal{F}$*, there exists* $b \in \mathbb{N}$ *such that for any derivation from* $\mathcal{F}$ *to* F_k *the core of* F_k *has treewidth at most* b*.*

This property can also be expressed in terms of models: a set of rules $\mathcal{R}$ is BTS if and only if there exists a (potentially infinite) universal model of the knowledge base whose treewidth is bounded.

Example 2.3 (Non-BTS)
Consider the following set of rules $\mathcal{R}$*:*

- $R_1 = q(x) \rightarrow p(x, y) \wedge q(y)$
- $R_2 = p(x, y) \wedge p(y, z) \rightarrow p(x, z)$

The set of rules $\mathcal{R}$ is not BTS. Indeed, the first rule creates an infinite path, while the second computes the transitive closure of this path. Consider for instance the set of facts restricted to a single atom $q(a)$. For any $k \in \mathbb{N}$, the set of facts obtained from $q(a)$ by all derivations of length at most k "encodes" a clique of size k (more precisely, its primal graph contains such a clique), hence its treewidth is at least k. Furthermore, it is not possible to suppress this clique by considering the core of the set of facts (actually, under reasonable assumptions on the forward chaining mechanism, the generated set of facts is itself a core). It follows that any universal model of this knowledge base has unbounded treewidth.

Obviously, any FES set of rules is also a BTS set of rules, indeed it suffices to consider the number of terms occurring in a finite universal model as the bound on its treewidth. The decidability relies on Courcelle's Theorem [Cou89], which states that for any logical fragment in which the existence of a model implies the existence of a bounded treewidth model, then the satisfiability of a property that can be expressed in monadic second-order logic is decidable. However this does not provide explicitly an algorithm to answer the entailment problem. Though, there exists an expressive subclass of BTS called Greedy Bounded Treewidth Set (GBTS), for which we do not give a formal definition. Since we will not study it in this thesis, see [BMRT11, RT14, Tho13] for more details. Intuitively, this class ensures that one can build a bounded width tree decomposition of the saturated set of facts greedily, and furthermore compute a finite structure that encodes this tree decomposition. We do not know yet if this class is concrete or abstract, i.e., whether the associated recognition problem is decidable; though, this class is very interesting since on the one hand, a practical algorithm is provided, and on the other hand, to our knowledge, it includes all but one concrete known rule classes belonging to BTS but not to FES.

For each of these abstract rule classes, various concrete rule subclasses have been exhibited. In the following, we detail the main ones, and Figure 2.1 pictures the relations between them: an upwards edge going from a rule class C to a rule class C' means that any set of rules in class C is also in class C'.

2.2 Finite Expansion Set

Regarding FES rule classes, most of them can be classified in two different subfamilies (Section 3.2 is dedicated to them). The first family relies on how existential variables interact with each others, while the second one on how rules are triggered.

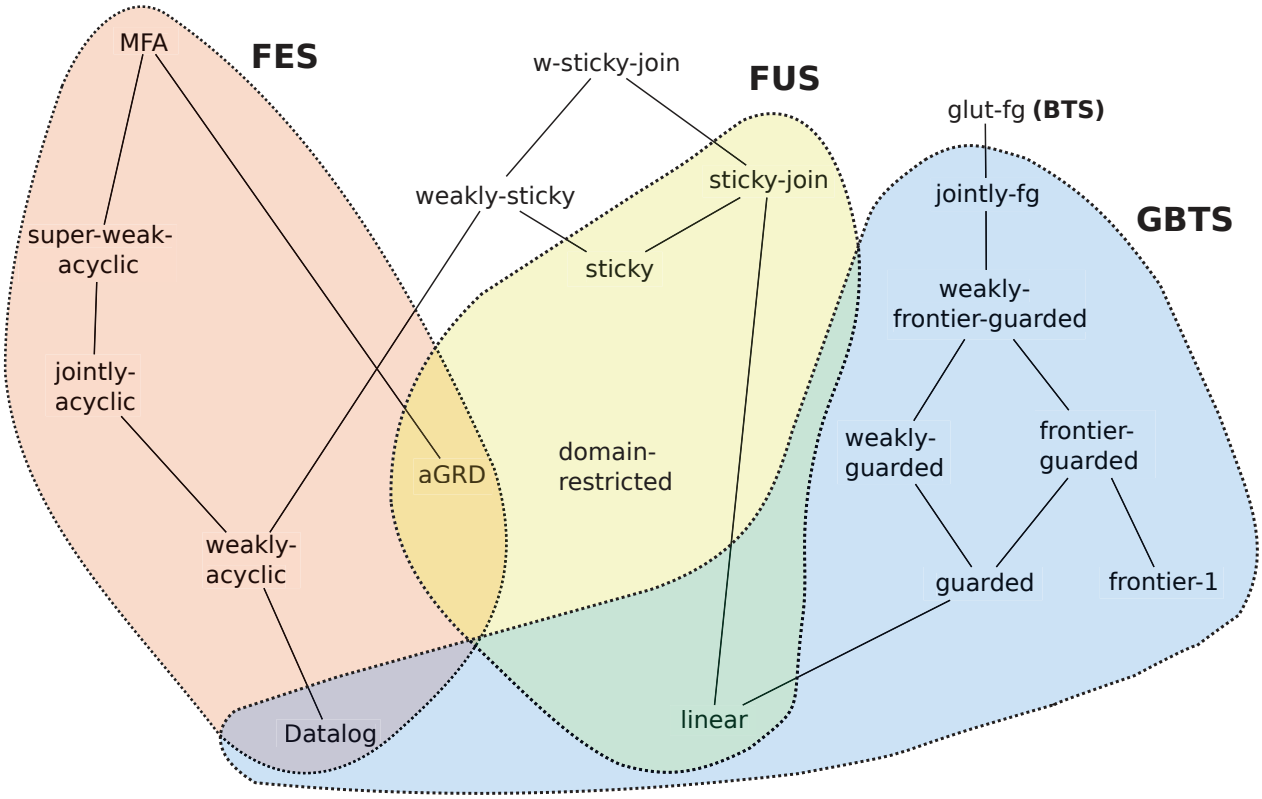

Figure 2.1: Relations between known decidable rule classes

In the first family, the simplest of all classes is that of range-restricted, or Datalog rules (as it corresponds exactly to the rules in Datalog queries). In these rules, no existential variable appears.

Definition 2.4 (Range-restricted (rr) [AHV95])
A set of rules $\mathcal{R}$ is range-restricted *if no existential variable occurs in a rule from $\mathcal{R}$.*

This rule class, while it seems really simple, allows to already express various interesting ontological properties, such that symmetry, transitivity, ...

Example 2.4 (Range-restricted)
Consider the set of rules $\mathcal{R}$ composed of the following rules:

- $p(x, y) \wedge p(y, z) \rightarrow p(x, z)$
- $p(x, y) \rightarrow p(y, x)$

These rules assert that p is a transitive and symmetric binary relation. The set of rules $\mathcal{R}$ is range-restricted.

This class has been first extended to allow for existential variables while limiting the way they propagate during the forward chaining, in order to avoid infinite creation of new variables. This constraint relies on a graph of predicate positions as defined below, where to ease the reading we denote by (p, i) the i^{th} position in predicate p.

Definition 2.5 (Predicate Position Graph)
Given a set of rules $\mathcal{R}$, its predicate position graph*, denoted by $PPG(\mathcal{R})$ is the directed labelled graph whose set of vertices is the set of predicate positions of $\mathcal{R}$. Then for each rule $R \in \mathcal{R}$ and each frontier variable x in B occurring in some position (p, i), edges with origin (p, i) are built as follows: there is an edge from (p, i) to each position (q, j) in H where x occurs, and there is a* special *edge from (p, i) to each position (q, j) in H where some existential variable y appears.*

Then we say that the *rank* of a predicate position (p, i) is *infinite* in a set of rules $\mathcal{R}$ if (p, i) belongs to a cycle going through a special edge in $PPG(\mathcal{R})$, and is *finite* otherwise.

The intuition behind this graph is that edges translate how terms can move from a position to another. Furthermore, its special edges mark the generation of existential variables, and if a cycle contains a special edge then an existential variable may lead to generate a new existential variable in the same position, hence may lead to infinitely many new existential variables.

Definition 2.6 (Weak-acyclicity (wa) [FKMP05])
A set of rules $\mathcal{R}$ is weakly-acyclic *if its predicate position graph does not contain any cycle going through a predicate position in which an existential variable occurs, i.e., if all predicate positions have finite rank.*

Example 2.5 (Weak-acyclicity)
Consider rules $R_1 = p(x, y) \rightarrow s(y, z)$ and $R_2 = s(x, y) \rightarrow p(y, x)$. The set of rules $\{R_1, R_2\}$ is weakly-acyclic as its predicate position graph does not contain any "special cycle" (i.e., a cycle going through a special edge), hence, all positions are of finite rank as can be seen on Figure 2.2.

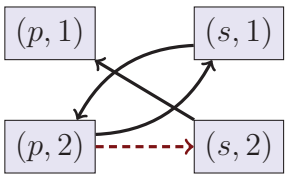

Figure 2.2: Predicate Position Graph of $\{R_1, R_2\}$ from Example 2.5

Weak-acyclicity has been further extended to joint-acyclicity by shifting the focus to existential variables instead of predicate positions [KR11]. This property relies on the notion of "move sets" attached to existential variables. To define them we use the following notation: given a set of atoms A and a term t, $Pos_A(t)$ is the set of predicate positions in the set A in which t occurs. Intuitively, the move set of a given existential variable contains all predicate positions in which it can appear, i.e., if some rule generates an existential variable z in some position (p, i), and another rule "moves" the variable in position (p, i) to a position (q, j), (as for instance for rule $p(x) \rightarrow q(x)$, with $i = j = 1$) then (q, j) is also in the move set of z.

The formal definition of a move set is given next.

Definition 2.7 (Move Set)
Given a set of rules $\mathcal{R}$, and an existential variable z occurring in some rule $R = (B, H) \in \mathcal{R}$, the move set *of z, denoted by $Move(z)$ is the smallest set of predicate positions such that:*

- $Pos_H(z) \subseteq Move(z)$,
- *for each rule $R' = (B', H')$ and each frontier variable y occurring in R', if $Pos_{B'}(y) \subseteq Move(z)$, then $Pos_{H'}(y) \subseteq Move(z)$.*

From this definition one can build another graph, called the joint-acyclicity graph.

Definition 2.8 (Joint-acyclicity Graph)
Given a set of rules $\mathcal{R}$, the joint-acyclicity graph *of $\mathcal{R}$, denoted by $JA(\mathcal{R})$ is the directed graph whose set of vertices is the set of all existential variables in $\mathcal{R}$, and where there is an edge from z_1 to z_2 whenever the rule $R = (B, H)$ that contains z_2 also contains a frontier variable y such that $Pos_B(y) \subseteq Move(z_1)$.*

In this graph, an edge from z_1 to z_2 intuitively means that variable z_1 may lead to create another existential variable z_2. Then a set of rules is jointly-acyclic if its joint-acyclicity graph contains no cycle, since the absence of such cycle ensures that no infinite "loop" of existential variable creation can happen.

Definition 2.9 (Joint-acyclicity [KR11])
A set of rules $\mathcal{R}$ is jointly-acyclic *if $JA(\mathcal{R})$ does not contain any cycle.*

Example 2.6 (ja $\not\subseteq$ wa)
Let $\mathcal{R}$ be the set composed of the following two rules:

- $R_1 = p(x_1, y_1) \rightarrow r(y_1, z_1)$,
- $R_2 = r(y_3, y_4) \wedge r(y_4, y_3) \rightarrow p(y_3, y_4)$.

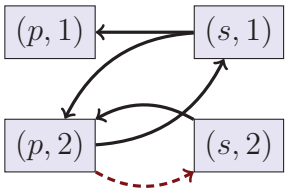

Figure 2.3: Predicate Position Graph of $\mathcal{R}$ from Example 2.6

Then one can check that $Move(z_1) = \{(r, 2)\}$ *and thus that* $JA(\mathcal{R})$ *does not contain any edge (hence no cycle), therefore* $\mathcal{R}$ *is jointly-acyclic.*

However, it can be observed that $\mathcal{R}$ *is not weakly-acyclic. Indeed its predicate position graph depicted in Figure 2.3 contains the cycle* $\{(p, 2), (r, 2)\}$ *that uses the special edge between those two positions.*

Joint-acyclicity can be generalised to super weak-acyclicity by looking at atom positions instead of predicate positions. We do not recall here its formal definition since it would require more technical notions, but this class and others are more deeply studied in Section 3.2, and all formal definitions can be found there.

The other subfamily relies on the notion of rule dependency, which intuitively, is a tool that allows to know if a rule may be triggered after another.

Definition 2.10 (Rule Dependency [Bag04, GHK$^+$13])
Given two rules $R_1 = B_1 \rightarrow H_1$ *and* $R_2 = B_2 \rightarrow H_2$, R_2 *depends on* R_1 *if there exists a set of facts* $\mathcal{F}$ *such that:*

(i) *there is a homomorphism* π_1 *from* B_1 *to* $\mathcal{F}$,

(ii) *there is a homomorphism* π_2 *from* B_2 *to* $\mathcal{F}' = \alpha(\mathcal{F}, R_1, \pi_1)$,

(iii) π_2 *is not a homomorphism from* B_2 *to* $\mathcal{F}$,

(iv) $\alpha(\mathcal{F}', R_2, \pi_2) \neq \mathcal{F}'$.

Hence, R_2 depends on R_1 if there exists a set of facts $\mathcal{F}$ such that R_1 can be applied to $\mathcal{F}$ (point (i)) which yields a set of facts $\mathcal{F}'$ such that R_2 can be applied to $\mathcal{F}'$ (point (ii)), in a "useful" way (point (iv)), i.e., leading to actually add new atoms, and moreover, R_2 could not be applied on $\mathcal{F}$ (point (iii)) according to this homomorphism.

This notion of dependency may seem too abstract to be computed, however piece-unifiers can be used to actually compute it without considering all sets of facts (more details are given in Section 3.2).

Then given a set of rules $\mathcal{R}$, one can build a directed graph whose vertices are the rules, and where there is an edge from R_1 to R_2 if R_2 depends on R_1 (i.e., if an application of R_1 may lead to trigger a new application of R_2). We call such a graph, the *Graph of Rule Dependencies* of $\mathcal{R}$.

Definition 2.11 (Acyclic Graph of Rule Dependencies (aGRD) [Bag04])
A set of rules $\mathcal{R}$ is aGRD if its graph of rule dependencies contains no cycle.

The following example shows that aGRD and wa are incomparable.

Example 2.7 (aGRD and wa are incomparable)
First, we observe that the weakly-acyclic set of rules from Example 2.5 is not aGRD, since its graph of rule dependencies depicted on Figure 2.4 does contain a cycle. Hence, wa $\not\subseteq$ aGRD.

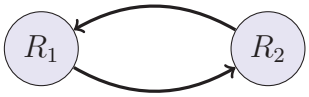

Figure 2.4: Graph of Rule Dependencies of $\{R_1, R_2\}$ from Example 2.5

Now, consider the set of rules composed of the following single rule $R = p(x, y) \wedge q(y) \rightarrow p(y, z)$. Rule R does not depend on itself. Hence, $\{R\}$ is aGRD. Indeed, this rule does not generate the atom $q(z)$ that would be necessary to apply it again. However its predicate position graph contains a special edge loop on position $(p, 2)$, and thus contains a position of infinite rank, therefore $\{R\}$ is not weakly-acyclic.

Two other rule classes that deserve to be mentioned are Model Faithful Acyclicity (MFA) and Model Summarizing Acyclicity (MSA), the latter one being an approximation of the first one at a lower complexity cost [GHK+13]. Both rely on effectively running a forward chaining algorithm (the skolem chase, see Section 3.1 for details) on a specific set of facts (called a critical instance), and with a modified set of rules. This critical instance has the nice property that if the forward chaining algorithm proposed in the paper halts on it, then it halts on any set of facts. Furthermore, after each rule application, the obtained set of facts is queried to check if some "cycle" has been found. Either the query finds a positive answer at some point, then the set of rules is not MFA (respectively MSA), or the forward chaining halts, then the set of rules is MFA (respectively MSA). These classes are precisely defined in Section 3.5.

2.3 Bounded Treewidth Set

Regarding BTS but not FES rules, the fundamental class is that of guarded rules [CGK08, CGK13].

Definition 2.12 (Guarded [CGK13])
A rule $R = (B, H)$ is guarded *if its body contains an atom α such that for any variable x occurring in B, x appears in α (such an atom is called a* guard*). By extension, a set of rules is* guarded *if all its rules are.*

Example 2.8 (Guarded)
Consider the rule $R = p(x, y) \wedge s(y) \rightarrow q(x, y, z)$, R is guarded since the atom $p(x, y)$ contains all variables from its body.

Another well-known BTS class is that of rules that have a frontier of size one.

Definition 2.13 (Frontier-one (fr1) [BLMS11])
A rule R is frontier-one *if $|fr(R)| = 1$. By extension, a set of rules is* frontier-one *if all its rules are.*

To any derivation from a set of facts $\mathcal{F}$ with a set of guarded and/or frontier-one rules $\mathcal{R}$ can be associated a tree decomposition of width $|terms(\mathcal{F})| + h$, where h is the maximal number of terms in the head of a rule from $\mathcal{R}$. Hence these classes are BTS. Furthermore, they are GBTS since such a tree decomposition can be built in a greedy way [BMRT11].

Example 2.9 (Guarded and Frontier-one are incomparable)
First, note that the guarded rule R from Example 2.8 is not frontier-one. Hence guarded $\nsubseteq$ fr1. Now consider the rule $R' = p(x_1, y) \wedge s(x_2, y) \rightarrow q(y, z)$. Rule R' is frontier-one, but is not guarded, since no atom contains all variables from its body. Hence fr1 $\nsubseteq$ guarded.

The guarded family has been widely studied, and a first generalisation of both guarded and frontier-one rules is the class of frontier-guarded rules, where only the frontier needs to be guarded. Indeed, the structure of the set of facts generated through forward chaining only depends on the way the frontier of rules is mapped during the forward chaining. Therefore, to ensure the (G)BTS property, only the frontier of each rule needs to be guarded.

Definition 2.14 (Frontier-guarded (fg) [BLM10, BMRT11])
A rule $R = (B, H)$ is guarded *if its body contains an atom α such that for any variable x occurring in the frontier of R, x appears in α. By extension, a set of rules is* frontier-guarded *if all its rules are.*

Example 2.10 (Frontier-guarded)
Consider the following three rules:

- $R_1 = q(x_1, x_2) \rightarrow p(x_1, z_1, z_2)$
- $R_2 = p(x_1, x_2, x_3) \rightarrow q(z_1, z_2) \wedge r(z_1, x_3)$
- $R_3 = r(x_2, x_1) \wedge q(x_1, x_3) \rightarrow s(x_2)$

The set of rules $\{R_1, R_2, R_3\}$ is frontier-guarded, indeed, the first two rules are guarded since they contain only a single atom. Regarding the third rule, the only frontier variable is x_2, and therefore atom $r(x_2, x_1)$ guards all frontier variables.

Guarded rules have been extended in another direction, by restricting the guard to variables that are "affected". Intuitively, an affected predicate position is a position in which some existential variable may be generated during the forward chaining, and an affected variable is a variable that occurs in some affected position. This generalisation has been called weakly-guarded [CGK13].

The idea behind this rule class, is that if a predicate position is not affected, then it is not dangerous for the construction of a tree decomposition of bounded width, since only constants (and variables already present in the initial set of facts if any) may occur in this position.

Definition 2.15 (Affected Position and Variable)
Given a set of rules $\mathcal{R}$, a predicate position (p, i) is affected*, if there exists an existential variable occurring in position (p, i) or if there is a frontier variable x on position (p, i) in the head of a rule occurring in an affected position in its body.*

A variable from a rule body is said to be affected *if it occurs in some affected position.*

Definition 2.16 (Weakly-guarded (wg) [CGK08, CGK13])
A set of rules $\mathcal{R}$ is weakly-guarded *if for any rule $R = (B, H) \in \mathcal{R}$, its body contains an atom α such that for any affected variable x occurring in B, x appears in α.*

Example 2.11 (fg and wg are incomparable)
First, notice that the set of rules from Example 2.10 is not weakly-guarded, since in rule R_3, all x_1, x_2 and x_3 are affected, and no atom contains all of them. Indeed, positions $(p, 2)$ and $(p, 3)$ are affected because of rule R_1, and positions $(q, 1), (q, 2)$ and $(r, 1)$ because of rule R_2. Furthermore, in rule R_3, x_1 occurs in position $(q, 1)$, x_2 in position $(r, 1)$ and x_3 in position $(q, 2)$. Hence, fg $\not\subseteq$ wg.

Now, consider rules $R_4 = p(x) \to q(x, z)$ and $R_5 = r(y, x_1) \wedge q(y, x_2) \to s(x_1, x_2)$. The set of rules $\{R_4, R_5\}$ is not frontier-guarded since rule R_5 is not. However it is weakly-guarded, since its set of affected positions is $\{(q, 2), (s, 2)\}$ and atom $q(y, x_2)$ in R_5 guards x_2, the only variable occurring in an affected position. Hence, wg $\not\subseteq$ fg.

Both of these generalisations of guardedness can be combined together, defining a more general rule class: namely, weakly-frontier-guarded.

Definition 2.17 (Weakly-frontier-guarded (wfg) [BMRT11])
A set of rules $\mathcal{R}$ is weakly-frontier-guarded *if for any rule $R = (B, H) \in \mathcal{R}$, its body contains an atom α such that for any affected variable x occurring in the frontier of R, x appears in α.*

Example 2.12 (Weakly-frontier-guarded)
Consider the rules $R_1 = p(x) \to q(x, z)$ and $R_2 = r(y, x_1) \wedge q(y, x_2) \wedge q(y, x_3) \to s(x_1, x_2)$ The set of rules $\{R_1, R_2\}$ is not weakly-guarded since position $(q, 2)$ is affected and thus rule R_2 would need to have an atom guarding both x_2 and x_3. It is

not frontier-guarded either, since both x_1 and x_2 occur in the frontier of R_2, and no atom contains both of them. However it is weakly-frontier-guarded as atom $q(y, x_2)$ guards x_2, the only affected variable occurring in the frontier.

This rule class has been further generalised by refining the notion of affected variable thanks to the use of the move set of a given variable. Roughly, the same idea of propagation is used, but the condition on the propagation is more accurate.

Definition 2.18 (Jointly-affected Position and Variable)
Given a set of rules $\mathcal{R}$, a predicate position (p, i) is a jointly-affected position *if there is an existential variable z such that $(p, i) \in Move(z)$.*

A variable x from a rule body is a jointly-affected variable *if it occurs only in jointly-affected positions.*

Definition 2.19 (Jointly-weakly-frontier-guarded (j-wfg) [KR11])
A set of rules $\mathcal{R}$ is jointly-weakly-frontier-guarded if for any rule $R = (B, H) \in \mathcal{R}$, its body contains an atom α such that for any jointly-affected variable x occurring in the frontier of R, x appears in α.

Example 2.13 (Jointly-weakly-frontier-guarded)
Consider the following two rules:

- $R_1 = p(x) \rightarrow q(x, z)$
- $R_2 = r(x, x_1) \wedge q(y, x_1) \wedge q(y, x_2) \rightarrow s(x_1, x_2)$

The set of rules $\{R_1, R_2\}$ is not weakly-frontier-guarded, indeed, position $(q, 2)$ is affected, and the frontier of R_2 is $\{x_1, x_2\}$. Both variables occur in position $(q, 2)$ and thus should be guarded, however no atom contains both of them.

However, it is jointly-weakly-frontier-guarded, since $Move(z) = \{(q, 2)\}$ and thus the only jointly-affected variable in R_2 is x_2 (since x_1 also occurs in the non jointly-affected position $(r, 2)$). Therefore, atom $q(y, x_2)$ guards all jointly-affected variables.

Interestingly, all those rule classes are not only BTS but also GBTS, and to the best of our knowledge, only a single rule class is known to be BTS but neither FES nor GBTS, namely glut-frontier-guarded defined next.

Definition 2.20 (Glut Variable)
Given a set of rules $\mathcal{R}$ and a variable x, x is a glut variable *if there exists an existential variable z such that z occurs in some cycle in $JA(\mathcal{R})$ and x appears in a position in $Move(z)$.*

Once again, the idea behind this rule class is to guard only dangerous variables, and the notion of glut variable refines that of jointly-affected variable by furthermore looking at the joint-acyclicity graph of the set of rules to determine if the suspicious variable occurs in a position where an existential variable that belongs to a dangerous cycle may also occur.

Definition 2.21 (Glut-frontier-guarded (glut-fg) [KR11])
A set of rules $\mathcal{R}$ is glut-frontier-guarded *if for any rule $R = (B, H) \in \mathcal{R}$, its body contains an atom α such that for any glut variable x occurring in the frontier of R, x appears in α.*

Example 2.14 (Glut-frontier-guarded)
Consider the set of rules $\mathcal{R}$ composed of the following two rules:

- $p(x) \rightarrow q(x, z)$
- $q(y, x_1) \wedge q(y, x_2) \rightarrow s(x_1, x_2)$

The set $\mathcal{R}$ is not jointly-weakly-frontier-guarded since $Move(z)$ contains position $(q, 2)$, and thus in rule R_2 frontier variables x_1 and x_2 occur only at jointly-affected positions, and hence need to be guarded.

On the other hand, $JA(\mathcal{R})$ contains no edge, and thus no cycle. Hence, no variable from $\mathcal{R}$ is a glut variable, therefore, $\mathcal{R}$ is glut-frontier-guarded.

2.4 Finite Unification Set

Regarding FUS classes, the first one we recall is a specialisation of guarded rules that enjoys the FUS property: namely, the linear rules [CGK08], they are called "atomic-hypothesis" in [BLMS09].

Definition 2.22 (Linear [CGK08, BLMS09])
A rule $R = (B, H)$ is linear *if $|B| = 1$. By extension, a set of rules is* linear *if all its rules are.*

While linear rules may seem quite inexpressive, they are actually enough to express many common ontological properties such as subclass, domain, range, inverse relations, ... Furthermore, Section 4.2 is devoted to combining linear rules with transitivity rules and more details are given there.

Two other FUS classes that are incomparable both with linear and with each other are sticky and domain-restricted rules. Both classes impose constraints that can be understood by considering a query rewriting step. The direct rewriting of a query Q with a rule R involves the creation of a new variable if some variable from the body does not occur in the unified atoms from the head. Since not all atoms from a rule head may be part of the unification, by imposing that all variables from a rule body occur in all atoms from a rule head, we are sure that no new variable is generated during query rewriting. The sticky and domain-restricted classes relax this constraint by allowing the generation of new variables, but controlling their occurrences in order to ensure the finiteness of the query rewriting process. Note that linear and sticky rules have been generalised by an abstract property named "backward shyness" [Tho13], which roughly says that a variable generated during a query rewriting sequence does not occur in two atoms of the obtained rewriting.

The sticky rule class relies on the notion of marked variable set.

Definition 2.23 (Marked Variable Set)
Given a set of rules $\mathcal{R}$, the marked variable set *of $\mathcal{R}$ is built by the following marking procedure.*

(i) for each rule in $\mathcal{R}$ and for each variable v occurring in its body, if v does not occur in all its head atoms, mark (each occurrence of) v in its body,

(ii) apply until fixpoint: for each rule R_i, if a marked variable v appears in the k^{th} position of an atom using predicate p in its body, then for each rule R_j (including $i = j$), and for each variable x occurring in the k^{th} position of an atom using predicate p in the head of R_j, mark (each occurrence of) x in the body of R_j.

Definition 2.24 (Sticky [CGP10a])
A set of rules $\mathcal{R}$ is sticky *if given the marked variable set of $\mathcal{R}$, each marked variable occurs at most once in a rule body.*

Definition 2.25 (Domain-restricted (dr) [BLM10])
A rule $R = (B, H)$ is domain-restricted *if every atom $h \in H$, either h contains no variable from B, or it contains them all. By extension, a set of rules is* domain-restricted *if all its rules are.*

The idea behind the domain-restricted rule class is that for each rule either no new variable is generated during the rewriting step, or these new variables are "disconnected" from the others.

Example 2.15 (sticky $\not\subseteq$ dr)
Consider the set of rules composed of the following single rule: $R = p(x_1, y_1) \rightarrow q(y_1, z_1) \wedge s(x_1, z_1)$. This set of rules is sticky, since no marked variable occurs twice in a rule body (indeed, even if both x_1 and y_1 are marked, neither occurs twice in the rule body), but it is not domain-restricted as atom $q(y_1, z_1)$ contains the variable y_1 but not the variable x_1 (similarly, atom $s(x_1, z_1)$ contains the variable x_1 but not the variable y_1).

Example 2.16 (dr $\not\subseteq$ sticky)
Consider the set of rules composed of the following single rule: $R = p(x_1, y_1) \wedge p(y_1, x_1) \rightarrow s(x_1, y_1, z_1) \wedge q(z_1, z_2)$. This set of rules is domain-restricted, but is not sticky: indeed, variable x_1 is marked since it does not occur in atom $q(z_1, z_2)$ and occurs twice in the same rule body.

While these two rule classes are incomparable, an interesting intersection of them is that of rules where every atom in their head contains all the variables from the body. Indeed, they are obviously domain-restricted, and no variable is ever marked by the sticky marking procedure.

Then, several generalisations of sticky rules have been obtained by refining the way variables are marked. Weakly-sticky rules generalise both sticky and weakly-acyclic rules, while sticky-join rules generalise sticky and linear rules. Furthermore, those two rule classes are incomparable, but have a common generalisation, namely, weakly-sticky-join.

Definition 2.26 (Weakly-sticky (w-sticky) [CGP10b])
A set of rules $\mathcal{R}$ is weakly-sticky *if given the marked variable set of $\mathcal{R}$ and its predicate position graph, all marked variables that occur more than once in a rule body appear at some position of finite rank in this body.*

Example 2.17 (Weakly-sticky)
Consider the set of rules $\mathcal{R}$ composed of the following rules:

- $R_1 = q(x_1, x_2) \rightarrow p(x_2, x_1, z_1, z_2)$
- $R_2 = p(x_1, x_2, x_3, x_4) \rightarrow q(x_2, z_1) \wedge r(x_2, x_4)$
- $R_3 = r(x_1, x_2) \wedge q(x_1, x_3) \rightarrow s(x_2, x_3)$

This set of rules is not weakly-acyclic: indeed, one can see on Figure 2.5 that positions $(q, 2)$ and $(p, 4)$ are of infinite rank.

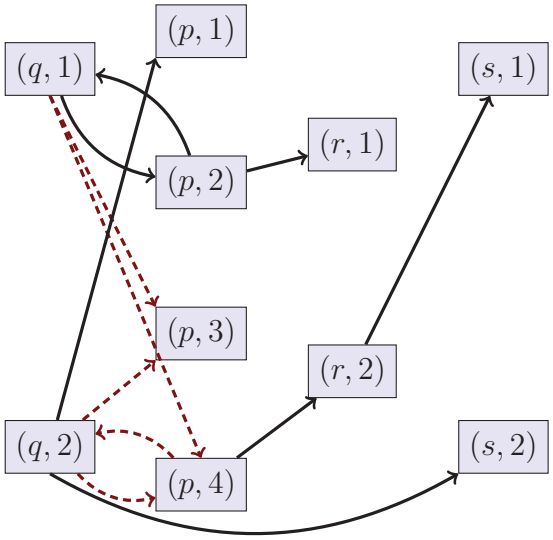

Figure 2.5: Predicate Position Graph of $\mathcal{R}$ from Example 2.17

Furthermore it is not sticky as variable x_1 in rule R_3 is marked in the first step of the marking procedure and occurs twice in the body of this rule.

However it is weakly-sticky: indeed even if the variable x_1 is marked in rule R_3, it occurs in position $(r, 1)$ which is of finite rank (note that x_1 also occurs in position $(q, 1)$ which is also of finite rank, though, a single finite rank position is enough).

Defining sticky-join sets of rules requires more notions.

Definition 2.27 (Sticky Applicability)
Given two rules $R_1 = (B_1, H_1)$ and $R_2 = (B_2, H_2)$, we say that R_1 is sticky applicable *to R_2 if there exist atoms $a \in H_1$ and $b \in B_2$ such that a and b unify (i.e., there exists a piece-unifier of $\{b\}$ with the rule $B_1 \to a$),*

From there, we can define the notion of an expanded set of rules.

Definition 2.28 (Expanded Set of Rules)
Given a set of rules $\mathcal{R}$, its expanded set $\mathcal{R}^*$ is obtained as follows. Initially, for each rule $R \in \mathcal{R}$, the rule R labelled by $\emptyset$ is added to $\mathcal{R}^*$. Then we apply the following step until a fixpoint is reached: for each pair of rules $(R = (B, H), R' = (B', H')) \in \mathcal{R} \times \mathcal{R}^*$ (including the case $R = R'$), if R is sticky applicable to R' due to atoms $a \in H$ and $b \in B'$, then let R^+ be the rule $\sigma_\mu(B) \to \sigma_\mu(a)$, where μ is the piece-unifier for a and b. If $\mathcal{R}^*$ already contains a labelled rule R'' isomorphic to R^+, then the pair (R', b) is added to the label set of R''; otherwise the rule R^+ labelled by $\{(R', b)\}$ is added to $\mathcal{R}^*$. Note that each time the above step is applied, we assume (as usual) that the two rules do not share any variable name.

Then, the marking procedure used for sticky set of rules can be improved as follows.

Definition 2.29 (Expanded Marked Variable Set)
Given a set of rules $\mathcal{R}$, the expanded marked variable set *of $\mathcal{R}$ is built by the following marking procedure. Let $\mathcal{R}^*$ be the expanded set of $\mathcal{R}$. For each rule $R = (B, H) \in R^*$ and each non-frontier variable x occurring in B, (each occurrence of) x in B is marked. Then the following step is applied until fixpoint. For each rule $R = (B, H) \in \mathcal{R}^*$, and for each pair $(R' = (B', H'), \alpha)$ in the label set of R, if a universal variable x occurs in H at positions $p_1, \ldots, p_k$ (with $k \geq 1$), and at each position $p_1, \ldots, p_k$ in $\alpha \in B'$ a marked variable occurs, then (each occurrence of) x in B is marked.*

Thanks to this expanded marked variable set, we are now ready to define the sticky-join class of rules.

Definition 2.30 (Sticky-join (sticky-j) [CGP10b])
A set of rules $\mathcal{R}$ is sticky-join *if given its expanded marked variable set, each marked variable occurs at most once in a rule body.*

In the same way as sticky can be generalised to weakly-sticky, sticky-join can be generalised to weakly-sticky-join, which yields a common generalisation of the whole sticky family.

Definition 2.31 (Weakly-sticky-join (w-sticky-j) [CGP10b])
A set of rules $\mathcal{R}$ is weakly-sticky-join *if given the expanded marked variable set of $\mathcal{R}$ and its predicate position graph, all marked variables that occur more than once in a rule body appear at some position of finite rank.*

It should be pointed out that weakly-sticky rules (hence, their generalisation weakly-sticky-join rules) are neither BTS nor FUS, though they are provided with an algorithm able to handle them to solve the entailment problem [CGP10b].

Finally, the last rule class we define here is that of disconnected rules, which are GBTS, FES and FUS.

Definition 2.32 (Disconnected (disc) [BLM10])
A rule $R = (B, H)$ is disconnected *if $var(B) \cap var(H) = \emptyset$. By extension, a set of rules is* disconnected *if all its rules are.*

All those rule classes fulfil various properties that are for the most part incomparable, however they can be combined thanks to the graph of rule dependencies as stated by the next theorem. We recall that an edge from R_1 to R_2 means that R_2 depends on R_1.

Theorem 2.1 (Combined Rule Classes [BLMS09, BLMS11])

Let $\mathcal{R}$ be a set of rules. Conjunctive query entailment is decidable if there exists a tri-partition $(S_{FES}, S_{BTS}, S_{FUS})$ of the strongly connected components (S.C.C.) of the graph of rule dependencies of $\mathcal{R}$ such that there is no edge from a rule in a S.C.C. in S_{FUS} or S_{BTS} to a rule from another S.C.C. in S_{FES} or S_{BTS}.

The above theorem means that we can safely combine rule classes provided that it is possible to first apply all FES rules to $\mathcal{F}$ thanks to some forward chaining algorithm which yields a saturated set of facts $\mathcal{F}^*$, and on the other hand rewrite the query with the FUS rules obtaining a rewriting set $\mathcal{Q}$, and finally use a BTS algorithm with $\mathcal{F}^*$ and $\mathcal{Q}$ on the subset of rules that satisfies the BTS property.

Example 2.18 (Combining Rule Classes)
Let $\mathcal{R}$ be the set of rules composed of the following rules:

- $R_0 = p(x, y) \wedge p(y, z) \rightarrow p(x, z)$
- $R_1 = p(x, y) \rightarrow p(y, x) \wedge q(x, y, x)$
- $R_2 = r(x, y) \rightarrow q(x, z, t)$
- $R_3 = q(x, y, z) \rightarrow r(z, t)$

- $R_4 = p(x, x) \wedge r(y, u) \wedge s(t, y, x) \rightarrow s(z, z, z)$
- $R_5 = s(x, x, x) \wedge s(y, y, y) \rightarrow s(z, x, y)$

The set of rules $\mathcal{R}$ does not belong to any concrete decidable class. As shown in Figure 2.6, the graph of rule dependencies of $\mathcal{R}$ contains three strongly connected components. Each one satisfies some concrete properties : C_0 is range-restricted, hence FES; C_1 is linear and thus FUS and BTS (actually GBTS); C_2 is domain-restricted therefore FUS.

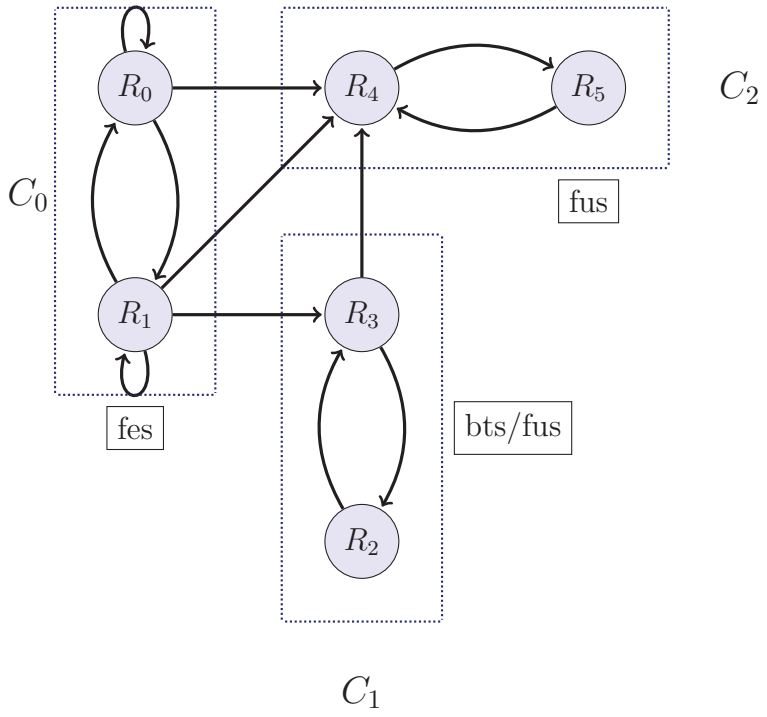

Figure 2.6: Graph of Rule Dependencies of $\mathcal{R}$ from Example 2.18

There are two interesting different safe combinations of these SCC, which differ on how C_1 is considered (obviously, C_0 could also be considered as BTS since any set of rules satisfying the FES property is also a BTS). If C_1 is considered as a BTS, then one could first apply all rules from C_0 until fixpoint in order to obtain a set of facts $\mathcal{F}^*$, then rewrite C_2 until fixpoint to obtain a UCQ $\mathcal{Q}$, then, use some BTS algorithm with input knowledge base $(\mathcal{F}^*, C_1)$ and query $\mathcal{Q}$. Otherwise, if C_1 is considered as FUS, then the query can be first rewritten according to $C_1 \cup C_2$ and then evaluated against $\mathcal{F}^*$.

One way or the other, CQ entailment over any knowledge base using this set of rules is decidable.

Note that weakly-sticky and weakly-sticky-join (defined previously) are actually another way to combine paradigms (in this case, FES and FUS).

We conclude the presentation of these rule classes by recalling the complexities of the CQ entailment problem over knowledge bases where rules belong to some known decidable rule class (Table 2.1). As usual, two different measures of complexity are considered: the combined complexity, where the input contains the set of rules, the set of facts and the query; and the data complexity, where the input contains only the set of facts since the set of rules and the query are assumed to be fixed. This complexity is often considered more relevant because the query and the rules are usually far smaller than the set of facts in practical applications. However, both complexities can help to understand where the difficulties lie. Indeed, for instance, guarded and weakly-guarded rules have both the same (worst-case) combined complexity, however, entailment over guarded rules can be solved in $PTime$ in data complexity, while it is *ExpTime-hard* over weakly-guarded rules. On the other hand, all FUS classes are by definition in AC_0 in data complexity (the same complexity as conjunctive query entailment without rule), while ranging from $PSpace$ for linear rules to $ExpTime$ for sticky-join rules in combined complexity. Finally, note that for some classes the complexity bounds are not tight (combined complexity for aGRD as well as data complexity for glut-fg, and both for MSA and MFA).

2.5 Description Logics

Description Logics (DLs) are the prominent family of formalisms to represent and do reasoning with ontologies [BCM$^+$03].We will not provide here a detailed presentation of DLs but rather point out their relationships with the existential rule framework.

In description logics, a knowledge base is composed of an ABox (for "assertion" component), which is a set of ground facts; and a TBox (for "terminological" component), which is a set of axioms. An axiom is a logical statement which expresses an inclusion between concepts (or classes) or roles (or properties). With a specific description logic is associated a set of operators (or constructors) which allows to build complex concepts and roles from atomic ones. The expressivity of a description logic depends on the set of allowed operators and the form of the concepts and roles that may occur on each side of the inclusion.

The semantics of a description logic knowledge base can be given by a translation into first-order logic. To each atomic concept is assigned a unary predicate and to each atomic role a binary predicate. Concept inclusions of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are (possibly complex) concepts are logically translated into formulae of the form $\forall x(\phi_{C_1}(x) \rightarrow \phi_{C_2}(x))$, where $\phi_C(x)$ is the logical translation of C, i.e., $C(x)$ if C is an atomic concept, otherwise it is more generally a formula with a single free variable x. Similarity, role inclusions of the form $r_1 \sqsubseteq r_2$, where r_1 and r_2 are (possibly complex) roles, are logical translated into formulae of the form $\forall x \forall y(\phi_{r_1}(x, y) \rightarrow \phi_{r_2}(x, y))$, where $\phi_r(x, y)$ is the logical translation of r, i.e., $r(x, y)$

Rule Class	Combined	Data
agrd	?	AC_0
fg	2*ExpTime-c* [BMRT11]	*PTime-c* [BMRT11]
fr-1	2*ExpTime-c* [BMRT11]	*PTime-c* [BMRT11]
guarded	2*ExpTime-c* [CGK08]	*PTime-c* [CGL09]
gbts	2*ExpTime-c* [BMRT11, Tho13, BMRT14]	*ExpTime-c* [BMRT11]
glut-fg	3*ExpTime-c* [KR11]	$\geq$*ExpTime*
ja	2*ExpTime-c* [KR11]	*PTime-c* [KR11]
j-fg	2*ExpTime-c* [KR11]	*ExpTime-c* [KR11]
linear	*PSpace-c* [CGL10a]	AC_0 [CGL09]
msa	2*ExpTime-c* [ZZY15]	*PTime-c* [Mar09]
mfa	2*ExpTime-c* [ZZY15]	*PTime-c* [Mar09]
rr	*ExpTime-c* [CLM81]	*PTime-c* [DEGV01]
sticky	*ExpTime-c* [CGP10a]	AC_0 [CGP10a]
sticky-j	*ExpTime-c* [CGP10b]	AC_0 [CGP10b]
swa	2*ExpTime-c* [Mar09]	*PTime-c* [Mar09]
wa	2*ExpTime-c* [FKMP05, CGP10b]	*PTime-c* [FKMP05, DEGV01]
w-fg	2*ExpTime-c* [BMRT11]	*ExpTime-c* [BMRT11]
wg	2*ExpTime-c* [CGK08]	*ExpTime-c* [CGL10a]
w-sticky	2*ExpTime-c* [CGP10b]	*PTime-c* [CGP10b]
w-sticky-j	2*ExpTime-c* [CGP10b]	*PTime-c* [CGP10b]

Table 2.1: Complexity of CQ entailment

if r is an atomic role, otherwise it is more generally a formula with exactly two free variables x and y.

An element of the ABox is of the form $C(a)$ where C is a concept or $r(a, b)$ where r is a role (and a and b are individuals, i.e., constants). To simplify the presentation we will assume that C and r are atomic, hence each element of the ABox can be seen as a ground atom.

Example 2.19 (Description Logics)
Let $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ where $\mathcal{A}$ is an ABox and $\mathcal{T}$ is a TBox. The ABox is composed of the following ground atoms: $\mathcal{A} = \{A(a), B(a), r(b, a)\}$. The TBox contains the following axioms:

- $A \sqsubseteq B$,
- $B \sqcap C \sqsubseteq \exists r.D$
- $\neg r \sqsubseteq s^-$
- $B \sqsubseteq \geq 2r.B$

The logical translation of $\mathcal{A}$ is $\mathcal{A}$ itself (where a and b are constants). The logical translation of $\mathcal{T}$ is:

- $\forall x(A(x) \rightarrow B(x))$
- $\forall x(B(x) \wedge C(x) \rightarrow \exists y(r(x,y) \wedge D(y)))$
- $\forall x \forall y(\neg r(x,y) \rightarrow s(y,x))$
- $\forall x(B(x) \rightarrow \exists y \exists z(r(x,y) \wedge r(x,z) \wedge y \neq z))$

General description logics and existential rules are incomparable with respect to expressivity. On the one hand, description logics allow for a variety of operators expressing disjunction, negation and cardinalities that are generally not translatable into the existential rule framework (e.g., axiom 3 from Example 2.19 is equivalent to the negative constraint $\forall x \forall y(r(x,y) \wedge s(y,x) \rightarrow \bot)$), while axiom 4 is not expressible). On the other hand, most description logics are restricted to unary and binary predicates. Furthermore, the axioms can only express "tree-shape" binary relations among variables, while in existential rules there is no constraint on the interactions between variables. For instance the rule $p(x,y) \wedge q(x,z) \rightarrow p(y,z)$ is not expressible in description logics.

Since the pioneer work on the CARIN language which combined description logics and positive range-restricted rules (i.e., Datalog rules) and was the first to mention the problem of answering conjunctive queries [LR96], several approaches have been developed to combine description logics and different kinds of (positive) rules. One can cite in particular DL-safe rules [MSS05], which ensure decidability of reasoning by restricting the application of rules to individuals from the ABox (in other words, variables in rules are "non-affected"); and nominal schemas [KR14] which can be used to encode DL-safe rules. Nominal schemas are concepts of the form $\{x\}$ which can be used in axioms. However, these extensions tend to lose the readable aspect of the description logics. Very recently, it has been shown that frontier-one rules under the restriction that their heads are "non-looping" (no cycle of length greater than 1 in the corresponding graph) can be combined with any description logic that can be expressed in GC^2 (the two-variable guarded fragment with counting quantifiers) while preserving decidability of the CQ entailment problem [AB15].

Historically, the work on description logics focused on the ontological part (the TBox) and queries were restricted to instance queries, i.e., ground atoms. These last ten years, the focus was shifted to dealing with databases and answering conjunctive queries. Since CQ answering appeared to be very complex with classical description logics, new description logics with a restricted use of operators have been introduced. These description logics are known as lightweight description logics. One major lightweight description logic family is the DL-lite family [CDL$^+$05, CDL$^+$07], a very simple DL designed for query answering on large databases. Another well-known description logic family is the $\mathcal{EL}$ family [Baa03] designed for dealing with very large

ontologies. These DLs, and more generally the so-called "Horn-DLs" correspond to fragments of existential rules (including possibly negative constraints to translate inclusions of the form $C_1 \sqsubseteq \neg C_2$ and equality rules to translate declarations of functional roles). In particular, DL-lite$_\mathcal{R}$ corresponds to a subclass of linear rules and $\mathcal{EL}$ (and $\mathcal{ELHI}$) to a subclass of frontier-guarded rules. More generally, Horn-DLs are translated into frontier-one rules when there is no role inclusion, otherwise into frontier-guarded rules.

2.6 Kiabora

To conclude this chapter, we mention that we have developed a software named Kiabora [LMR13], which given a set of rules as input provides different useful features: first, it is able to translate the set of rules to either single-piece headed rules or atomic-headed rules. More importantly, it is able to recognise most of the rule classes we have presented here, and it also tries to determine if the set of rules belongs to a decidable case by combining decidable rule classes via the graph of rule dependencies. It has been implemented in a way that allows one to easily add his own new decidable rule classes, and its source code is available under CeCILL (compatible with GPL) licensing. A simple web interface built on the first version of Kiabora is available on its website at `www.lirmm.fr/kiabora`. The second version of Kiabora, which is more efficient and considers further decidable classes has been integrated into the Graal toolkit (website: `http://graphik-team.github.io/graal/`).

Chapter 3

Acyclicity Conditions for Chase Termination

Contents

A mechanism taking into account a set of rules while querying data is that of forward chaining. It can be seen as a two-steps process: it first repeatedly applies rules to the set of facts, process also known as the *chase* in database theory; then it looks for an answer to the query in this saturated set of facts.

Intuitively, this mechanism can be seen as completing initial data by exploiting the knowledge described by the rules, until one can query the saturated data as a database.

This saturated set of facts encodes a universal model of the knowledge base, and since the problem of CQ entailment is undecidable, this process may not halt, that is, it may be the case that all universal models of the knowledge base are infinite.

Then a question naturally arises: can we know beforehand if the knowledge base has a finite universal model?

It appears that answering this question is also undecidable. However, various syntactic conditions on sets of rules that ensure the finiteness of a universal model have been exhibited, most of them relying on some notion of acyclicity. Informally, these conditions ensure that either at some point no rule can be applied while still adding some new knowledge, or that there is a bound on the number of fresh existential variables that need to be created.

In this chapter, we first define formally the notion of breadth-first derivation, from which follows a general definition of chase, then we study different chase variants, which differ in how they deal with redundancies potentially appearing in the sets of facts between each rule application (Section 3.1). We then examine in Section 3.2 the various existing acyclicity conditions which we group into two families: the "dependency-based" approach [Bag04, GHK$^+$13], and the "position-based" family which goes from weak-acyclicity [DEGV01] to super-weak-acylicity [Mar09]. In Section 3.3, we propose a new tool that allows to unify these different notions, and to extend them as detailed in Section 3.4. Finally, Section 3.5 is devoted to another kind of acyclicity conditions, that does not fit in any of these two families.

3.1 Different kinds of chase

In this section, we consider the different chase variants. Intuitively, given a knowledge base $(\mathcal{F}, \mathcal{R})$, all chase variants proceed in the following steps: first choose a rule $R = (B, H)$ in $\mathcal{R}$, then look for a homomorphism π from B to the current set of facts, and add $\pi^{safe}(H)$ to it provided that some conditions hold. We recall that π^{safe} is an extension of π that gives a new "name" to all existential variables occurring in H (see Definition 1.27).

These conditions are what distinguishes the different kinds of chase. For instance, in the *oblivious chase* (also called *naive chase*) [CGK08], a rule is applied according to some homomorphism π if it has not been already applied according to the exact same homomorphism.

This process can be slightly improved, in the sense that it can avoid some rule applications without compromising the construction of a universal model. If we consider two homomorphisms π and π' of the same rule body, they lead to add the "same" atoms if they map the frontier variables identically. In the *frontier-restricted chase* [BGMR14b], also known as *semi-oblivious chase* [GO13], a rule is not applied according to π if it has already been applied with a homomorphism π' that maps the frontier in the same way as π.

It already shows some distinctions with respect to finiteness; indeed if we consider

the rule $p(x, y) \rightarrow p(x, z)$ and the set of facts $\mathcal{F} = p(a, b)$, the oblivious chase adds infinitenely many atoms (as the rule can be infinitely applied according to a homomorphism that maps y differently each time), which yields the infinite set of facts $p(a, b) \wedge p(a, z_1) \wedge p(a, z_2) \wedge \ldots$. However in the case of the frontier-restricted chase, the rule is applied only once, since after that, the frontier is always mapped in the same way (x is mapped to a), hence the finite set of atoms $p(a, b) \wedge p(a, z_1)$ is obtained.

The *skolem chase* [Mar09] relies on a skolemisation of the rules: first, each rule R is transformed by replacing each occurrence of an existential variable z with a functional term $f_z^R(\vec{x})$, where $\vec{x}$ are the frontier variables of R. Then the oblivious chase is run on the skolemised rules. It can be easily checked that the skolem chase and the frontier-restricted chase yield isomorphic results (up to the renaming of the created existential variables by the corresponding skolem terms).

The *restricted chase* (which is also known as *standard chase*) [FKMP05] is a bit more involved. A rule $R = (B, H)$ is applied to $\mathcal{F}$ according to a homomorphism π if π cannot be extended to a homomorphism from $B \cup H$ to $\mathcal{F}$ (in which case we say that the application of R according to π is *useful*), otherwise $\mathcal{F} \cup \pi^{safe}(H)$ would be equivalent to $\mathcal{F}$. A rule such as $p(x) \rightarrow r(x, y) \wedge r(y, y) \wedge p(y)$ shows the distinction between the restricted chase and the frontier-restricted chase with respect to finiteness. Indeed, in the former case, the rule is applied only once since no further application is useful, while all applications map the frontier differently, and thus the frontier-restricted chase is infinite ($p(a) \wedge r(a, y_1) \wedge r(y_1, y_1) \wedge p(y_1) \ldots r(y_{k-1} \wedge r(y_k, y_k) \wedge p(y_k) \wedge \ldots$).

The *core chase* [DNR08] avoids the strongest possible form of redundancy: after each rule application it reduces the obtained set of facts by computing its core. Hence, it may remove atoms obtained from a previous rule application in a later step, as illustrated by the next example (Example 3.1). A nice property of the core chase is that if there exists a finite universal model of the knowledge base, then the core chase will halt.

While our contribution does not include a direct way to handle the core chase, we are able to consider the *equivalent chase*. This chase does not perform a rule application if the set of facts that would be obtained is equivalent to the current set of facts. In other words, if the core of the current set of facts is equal (i.e., isomorphic) to the obtained set of facts. We point out that the core chase is finite if and only if the equivalent chase is.

Example 3.1
Consider the set of facts $s(a)$, where a is a constant, and the set of rules composed of the following two rules:

- $R_1 = s(x) \rightarrow p(x, z)$
- $R_2 = s(x) \wedge p(x, y) \rightarrow p(x, x)$

In this example, we first apply R_1 according to the homomorphism $\{x \mapsto a\}$ and we create $F_1 = s(a) \wedge p(a, z_0)$. At this step, $Core(F_1) = F_1$. Note that we could not have applied R_2 before R_1. Then, we apply R_2 according to the homomorphism $\{x \mapsto a, y \mapsto z_0\}$, and we add the atom $p(a, a)$ and obtain $F_2 = s(a) \wedge p(a, z_0) \wedge p(a, a)$. The atom $p(a, z_0)$ is redundant with $p(a, a)$, and therefore, the core of F_2 is $Core(F_2) = s(a) \wedge p(a, a)$. Observe that the atom $p(a, z_0)$, which comes from the core of F_1, does not occur in the core of F_2, and thus has been "removed" at this step.

Our aim is to provide a general chase notion that is then specialised by each chase variant. This general chase is seen as a greedy process that adds more and more atoms to the initial set of facts. Hence it is intrinsically a monotonic process. That is why it does not include the core chase, but instead the equivalent chase which behaves in the same way with respect to finiteness.

As previously mentionned, all chase variants heavily rely on the notion of derivation, but not all derivations behave the same.

Indeed consider the following example.

Example 3.2
Let $\mathcal{F} = \{p(a, b)\}$ be a set of facts, $R_1 = p(x, y) \rightarrow p(y, z)$ and $R_2 = p(x, y) \rightarrow p(y, y)$ be two rules. One can see that applying rule R_1 over and over leads to an infinite derivation (therefore a universal model cannot be found since R_2 will never be applied) where at each rank k of the derivation, the following set of facts is obtained:

$$F_k = \{p(a, b), p(b, z_1), \ldots, p(z_{k-1}, z_k)\}$$

If both rules were to be applied, R_2 would create something more specific that anything we could generate with R_1. Indeed if we applied R_2 after a single application of R_1 we would obtain:

$$F_2 = \{p(a, b), p(b, z_1), p(b, b)\}$$

Then, further applications of rule R_1 would only add redundant knowledge.

The previous example shows that some derivations are more adapted to find a universal model than others, which leads to consider the notion of breadth-first derivation. A breadth-first derivation is obtained by considering at each "breadth-first step" all possible rule applications on the current set of facts, then applying them all, before moving to the next step, which will look for new applications on the resulting set of facts.

Definition 3.1 (Breadth-first Derivation)
Given $\mathcal{F}$ a set of facts and $\mathcal{R}$ a set of rules, a breadth-first derivation *of $\mathcal{F}$ w.r.t. $\mathcal{R}$ is a derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i = (B_i, H_i), \pi_i, F_i), \ldots$ such that for all $i \leq j \in dom(D)$, if $(F_i \setminus F_{i-1}) \cap \pi_j(B_j) \neq \emptyset$ then for all $k > j, \pi_k(B_k) \nsubseteq F_{i-1}$.*

The previous definition ensures that if at some point we use some atoms occurring for the first time in F_i to map a rule body, then we cannot use "only" F_{i-1} later. Intuitively, this ensures that once going to the next "breadth step", we cannot apply a rule that could have been applied in a previous breadth step according to the same homomorphism.

Definition 3.2 (Exhaustive Breadth-first Derivation)
Given $\mathcal{F}$ a set of facts, and $\mathcal{R}$ a set of rules, a (potentially infinite) breadth-first derivation $D = (\emptyset, \emptyset, \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ of $\mathcal{F}$ w.r.t. $\mathcal{R}$ is said to be exhaustive *if for all $i \in dom(D)$, for all $R = (B, H) \in \mathcal{R}$, and for all homomorphisms π from B to F_i there is some $k \in dom(D)$ such that $\pi_k = \pi$ and $R_k = R$.*

An exhaustive breadth-first derivation ensures that all rules have been applied according to all homomorphisms. It means that no other rule application can be found on any of the F_i. Note that the corresponding derivation may have to be infinite to satisfy this criterion.

Example 3.3
Consider the set of facts $\mathcal{F} = c(a)$ and the following rules:

- $R_1 = c(x) \rightarrow p(x, y) \wedge c(y)$
- $R_2 = c(x) \wedge p(x, y) \rightarrow p(x, x)$

An exhaustive breadth-first derivation for this example would be:

$\mathbf{i}$	$\mathbf{R_i}$	$\pi_\mathbf{i}$	$\mathbf{F_i}$
0	$\emptyset$	$\emptyset$	$\mathcal{F}$
1	R_1	$\{x \mapsto a\}$	$\mathcal{F} \cup \{p(a, y_1), c(y_1)\})$
2	R_2	$\{x \mapsto a, y \mapsto y_1\}$	$\mathcal{F} \cup \{p(a, y_1), c(y_1), p(a, a)\})$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$2k$	R_1	$\{x \mapsto y_{2k-1}\}$	$F_{2k-1} \cup \{p(y_{2k-1}, y_{2k}), c(y_{2k})\}$
$2k+1$	R_2	$\{x \mapsto y_{2k-1}, y \mapsto y_{2k}\}$	$F_{2k} \cup \{p(y_{2k-1}, y_{2k-1})\}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$

From the notion of exhaustive breadth-first derivation, one can strengthen Theorem 1.2 (soundness and completeness of derivations) as stated by the next proposition (which follows, e.g., from [BLMS11]). Intuitively, it ensures that if a Boolean conjunctive query is a logical consequence of the knowledge base, then any breadth-first derivation finds it in finitely many steps (while, of course, this may not be the case for all derivations).

Proposition 3.1

Let $\mathcal{F}$ be a set of facts, $\mathcal{R}$ be a set of rules, and Q be a Boolean conjunctive query. Let $D = (\emptyset, \emptyset, \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ be any exhaustive breadth-first derivation; then, $(\mathcal{F}, \mathcal{R}) \models Q$ iff there exists some $k \in \mathbb{N}$ such that $F_k \models Q$.

As already mentioned, the chase variants differ in how they "filter" the rule applications to consider (except for the core chase which simplifies the set of facts). In the following we unify these different variants with the notion of a derivation filter, that intuitively filters rule applications that will be preserved in the chase. Indeed, some of these applications may create redundant knowledge, and may be ignored.

Definition 3.3 (Derivation Filter)
Let $\mathcal{F}$ be a set of facts and $\mathcal{R}$ be a set of rules. A derivation filter *σ is a function that given a derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ of $\mathcal{F}$ w.r.t. $\mathcal{R}$ returns a sequence (with the same domain as D) of Boolean values such that*

- $\sigma(D)_0 = 1$;
- *if* $\sigma(D)_i = 0$ *then* $F_i \equiv F_{i-1}$
- *if* $\sigma(D)_i = 1$ *then there is some* $j < i$ *such that* $\pi_i(B_i) \subseteq F_j$ *and* $\sigma(D)_j = 1$.

We are now ready to define the *σ-chase*, which can be seen as the result of applying rules in a breadth-first manner, and ignoring some useless rule applications.

Definition 3.4 (σ-Chase)
Given $\mathcal{F}$ a set of facts, $\mathcal{R}$ a set of rules and σ a derivation filter, let D be any exhaustive breadth-first derivation of $\mathcal{F}$ w.r.t. $\mathcal{R}$. The σ-chase *of $\mathcal{F}$ w.r.t. $\mathcal{R}$ is the set of atoms defined as:*

$$\sigma\text{-}chase(\mathcal{F}, \mathcal{R}) = \bigcup_{i \in dom(D) \mid \sigma(D)_i = 1} F_i$$

Note that the above definition considers that the *σ-chase* of a given set of facts w.r.t. a given set of rules is unique. While it might not be the case, whatever the exhaustive breadth-first derivation one chooses, all possible σ-chases are equivalent and behave the same with respect to finiteness.

An important property of any chase variant built upon a given filter is that its result can be used to give all correct answers to a given query as stated by the next theorem.

Theorem 3.1

Let $\mathcal{F}$ be a set of facts, $\mathcal{R}$ be a set of rules, Q be a Boolean conjunctive query, and σ be a derivation filter.

$$(\mathcal{F}, \mathcal{R}) \models Q \Leftrightarrow \sigma\text{-}chase(\mathcal{F}, \mathcal{R}) \models Q$$

Proof: ($\Rightarrow$) Assume that $(\mathcal{F}, \mathcal{R}) \models Q$. Let $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ be an exhaustive breadth-first derivation of $\mathcal{F}$ w.r.t. $\mathcal{R}$. By Proposition 3.1 there exists $k \in \mathbb{N}$ such that $F_k \models Q$. Then, if $F_k \nsubseteq \sigma\text{-}chase(\mathcal{F}, \mathcal{R})$ it means that $\sigma(D)_k = 0$. In this case, there is some $j < k$ such that $F_k \equiv F_j$ and $\sigma(D)_j = 1$, thus there exists a homomorphism π from F_k to F_j. Furthermore, since $\sigma(D)_j = 1$, $F_j \subseteq \sigma\text{-}chase(\mathcal{F}, \mathcal{R})$. Therefore $\sigma\text{-}chase(\mathcal{F}, \mathcal{R}) \models Q$.

($\Leftarrow$) Suppose now that $\sigma\text{-}chase(\mathcal{F}, \mathcal{R}) \models Q$. Then, there exists a homomorphism π from Q to some finite part $\pi(Q)$ of $\sigma\text{-}chase(\mathcal{F}, \mathcal{R})$. From Definition 3.4, there is an exhaustive breadth-first derivation D from $\mathcal{F}$ to F_j such that $\pi(Q) \subseteq F_j$. We conclude with Proposition 3.1 that $(\mathcal{F}, \mathcal{R}) \models Q$. □

Obviously, the finiteness of the σ-*chase* and the computability of the i^{th} element of the sequence obtained from a filter (note that this is the case for all filters we present in this section) are enough to ensure one can compute it, therefore, guarantee decidability of the CQ entailment problem.

Proposition 3.2

Given a set of facts $\mathcal{F}$, a set of rules $\mathcal{R}$ and a derivation filter σ, if $\sigma\text{-}chase(\mathcal{F}, \mathcal{R})$ is finite and for any derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ and any $i \in dom(D)$, $\sigma(D)_i$ is computable, then $\sigma\text{-}chase(\mathcal{F}, \mathcal{R})$ is computable.

We now define several derivation filters that correspond to the chase variants we have introduced at the beginning of this section.

The equivalent filter considers only rule applications that lead to generate a set of facts that is not equivalent to the current set of facts.

Definition 3.5 (Equivalent Filter)
The equivalent filter*, denoted by σ_{equiv}, is the derivation filter defined for any derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ as: $\forall 0 < i \in dom(D)$,*

$$\sigma_{equiv}(D)_i = \begin{cases} 1 & if\ \mathcal{F}_{i-1} \not\equiv F_i \\ 0 & otherwise \end{cases}$$

The restricted filter checks redundancy locally as opposed to the global view of the equivalent filter.

Definition 3.6 (Restricted Filter)
The restricted filter*, denoted by $\sigma_{restricted}$, is the derivation filter defined for any derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ as: $\forall 0 < i \in dom(D)$,*

$$\sigma_{restricted}(D)_i = \begin{cases} 1 & if\ \pi_i\ cannot\ be\ extended\ to\ a\ homomorphism \\ & from\ B_i \cup H_i\ to\ F_{i-1} \\ 0 & otherwise \end{cases}$$

As already mentioned, the skolem chase has the same behaviour as the frontier-restricted chase, which considers only rule applications that map the frontier of the rule in different ways.

Definition 3.7 (Frontier-Restricted Filter)
The frontier-restricted filter*, denoted by σ_{fr}, is the derivation filter defined for any derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ as: $\forall 0 < i \in dom(D)$,*

$$\sigma_{fr}(D)_i = \begin{cases} 1 & if\ \forall j < i\ with\ R_i = R_j,\ \pi_j|_{fr(R_j)} \neq \pi_i|_{fr(R_i)} \\ 0 & otherwise \end{cases}$$

Finally, the oblivious filter consider all rules, and ignores only applications based upon a previously used homomorphism.

Definition 3.8 (Oblivious Filter)
The oblivious filter*, denoted by σ_{obl}, is the derivation filter defined for any derivation $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ as: $\forall 0 < i \in dom(D)$,*

$$\sigma_{obl}(D_i) = \begin{cases} 1 & if\ \forall j < i\ with\ R_i = R_j,\ \pi_j \neq \pi_i \\ 0 & otherwise \end{cases}$$

From these different filters, and since they all are sound and complete (indeed from Theorem 3.1, they all compute a set of facts that corresponds to a universal model), it is natural to ask whether they are comparable with respect to finiteness. To this end, we first define an order relation on the filters to be able to compare them.

Definition 3.9 (Filter Order Relation)
Given two derivation filters σ_1 and σ_2 we say that σ_1 is weaker *than σ_2, denoted by $\sigma_1 \preceq \sigma_2$, if for any set of rules $\mathcal{R}$ and set of facts $\mathcal{F}$, σ_1-chase$(\mathcal{F}, \mathcal{R})$ is finite implies that σ_2-chase$(\mathcal{F}, \mathcal{R})$ also is.*

Furthermore, we say that σ_1 is strictly weaker *than σ_2, denoted by $\sigma_1 \prec \sigma_2$ if $\sigma_1 \preceq \sigma_2$ and $\sigma_2 \not\preceq \sigma_1$.*

One can observe that the equivalent filter is the "strongest" of all possible chase variants as stated in Proposition 3.3: indeed, from its definition, it does not impose any restriction on which rule applications can be ignored but the equivalence between the sets of facts.

Proposition 3.3

The σ_{equiv} filter is maximal for the filter order relation $\preceq$.

Proof: Let σ be a derivation filter. Given a set of rules $\mathcal{R}$, assume that for some set of facts $\mathcal{F}$, σ-$chase(\mathcal{F}, \mathcal{R})$ is finite. Let $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_i, \pi_i, F_i), \ldots$ be an exhaustive breadth-first derivation of $\mathcal{F}$ w.r.t. $\mathcal{R}$. From the second point of the definition of a derivation filter (Definition 3.3), for any $i \in dom(D)$, if $\sigma(D)_i = 0$ then $F_i \equiv F_{i-1}$. Since for any j, $\sigma_{equiv}(D)_j = 0$ if and only if $F_j \equiv F_{j-1}$, $\sigma_{equiv}(D)_i = 0$. Then, for any i such that $\sigma(D)_i = 0$, $\sigma_{equiv}(D)_i$ is also equal to 0. This concludes the proof. □

Furthermore, the next proposition states that all filters defined above are comparable and even totally ordered with respect to the filter order relation.

Proposition 3.4

The following relations hold: $\sigma_{obl} \prec \sigma_{fr} \prec \sigma_{restricted} \prec \sigma_{equiv}$.

Proof: Inclusion between σ_{obl} and σ_{fr} follows immediately from their definitions, while the one between $\sigma_{restricted}$ and σ_{equiv} follows from Proposition 3.3. It is well-known that $\sigma_{fr} \preceq \sigma_{restricted}$, see, for instance [One13]. Examples 3.4 to 3.6 show that these inclusions are strict. □

Example 3.4 ($\sigma_{fr} \not\preceq \sigma_{obl}$)
Consider the set of rules $\mathcal{R} = \{p(x, y) \rightarrow p(x, z)\}$, *and the set of facts* $\mathcal{F} = \{p(a, b)\}$.

First, let us compute the oblivious chase, it is infinite, since all rule applications of the only rule use a homomorphism with a different image for the variable y. *Indeed in the first application it is mapped to* b, *then to the generated fresh variable* z_1, *then to* z_2, *and so on...*

$$\sigma_{obl}\text{-}chase(\mathcal{F}, \mathcal{R}) = \{p(a, b), p(a, z_1), \ldots, p(a, z_k), \ldots\}$$

While if we consider the frontier-restricted chase, all homomorphisms restricted to the single frontier variable x *are the same, and then, a single application is necessary.*

$$\sigma_{fr}\text{-}chase(\mathcal{F}, \mathcal{R}) = \{p(a, b), p(a, z_1)\}$$

Example 3.5 ($\sigma_{restricted} \not\preceq \sigma_{fr}$)
Consider the set of rules $\mathcal{R} = \{p(x) \rightarrow r(x, y) \wedge r(y, y) \wedge p(y)\}$, *and the set of facts* $\mathcal{F} = \{p(a)\}$.

In this example, the frontier is differently mapped for each rule application. Thus, the frontier-restricted chase (and the skolem chase since they are equivalent), is infinite.

$$\sigma_{fr}\text{-}chase(\mathcal{F}, \mathcal{R}) = \{p(a), r(a, y_1), r(y_1, y_1), p(y_1) \ldots, r(y_{k-1}, y_k), r(y_k, y_k), p(y_k) \ldots\}$$

However the generated set of facts is quite redundant: indeed, the "loop" $p(y, y)$ when applied to the fresh variable y_1 leads to something more specific than any other applications. Therefore the restricted chase is finite.

$$\sigma_{restricted}\text{-}chase(\mathcal{F}, \mathcal{R}) = \{p(a), r(a, y_1), r(y_1, y_1), p(y_1)\}$$

Example 3.6 ($\sigma_{equiv} \not\preceq \sigma_{restricted}$)
Consider the set of facts $\mathcal{F} = q(a)$ and the set of rules $\mathcal{R}$ composed of the following rules:

- $R_1 = q(x) \rightarrow r(x, y) \wedge p(x, z)$
- $R_2 = p(x, y) \rightarrow p(y, z)$
- $R_3 = r(x, y) \rightarrow p(x, x)$

In this example, the restricted chase first applies the first rule according to $\pi_1 = \{x \mapsto a\}$ and adds the atoms $p(a, z_1)$ and $r(a, y_1)$ to $F_0 = \mathcal{F}$ yielding F_1.

Then rule R_2 can be applied to F_1 according to $\pi_2 = \{x \mapsto a, y \mapsto z_1\}$ adding $p(z_1, z_2)$ to F_1 yielding F_2.

Rule R_3 can be applied to F_2 with homomorphism $\pi_3 = \{x \mapsto a, y \mapsto y_1\}$ adding atom $p(a, a)$ and yielding $F_3 = q(a) \wedge p(a, z_1) \wedge r(a, y_1) \wedge p(z_1, z_2) \wedge p(a, a)$.

Now, rule R_2 can be applied again, this time according to homomorphism $\pi_4 = \{x \mapsto z_1, y \mapsto z_2\}$, indeed this homomorphism cannot be extended to $p(x, y) \wedge p(y, z)$, and thus we add the atom $p(z_2, z_3)$.

In the end, the restricted chase of $\mathcal{F}$ and $\mathcal{R}$ is infinite and is as follows:

$$q(a) \wedge r(a, y_1) \wedge p(a, a) \wedge p(a, z_1) \wedge p(z_1, z_2) \wedge \cdots \wedge p(z_k, z_{k+1}) \wedge \ldots$$

Now let us consider the equivalent chase instead. While F_1, F_2 and F_3 are the same as with the restricted chase, any rule application to F_3 yields an equivalent set of facts. Indeed, F_3 is already redundant, the homomorphism that maps z_1 and z_2 to a, and y_1 to itself maps F_3 to its subset $q(a) \wedge r(a, y_1) \wedge p(a, a)$, then every $p(z_i, z_{i+1})$ added by an application of R_2 is also redundant. Therefore we obtain the finite set of facts $\sigma_{equiv}\text{-}chase(\mathcal{F}, \mathcal{R}) = F_3$.

3.2 Acyclicity notions

In this section, we study the various acyclicity notions ensuring a finite chase that have been proposed in the literature. As a matter of fact, all these conditions ensure the finiteness of the frontier-restricted chase (or the skolem chase), except for one (aGRD) that also ensures the finiteness of the oblivious chase. These properties can be divided in two big families relying respectively on dependency-based and position-based conditions. The former contains properties that look at how rules are triggered, while the latter contains those that look at how existential variables are

propagated along predicate positions. Note that some classes of rules and associated notions have already been defined in Chapter 2, but we define them again to ease the reading.

From there, we propose a new tool unifying both approaches, and show that it can be used to extend the decidability recognition based on existing acyclicity conditions. Two other properties that also ensure chase termination, namely Model Faithful Acyclicity and Model Summarizing Acyclicity are neither dependency nor position-based. These properties are discussed in Section 3.5, where we show that the second one can be integrated into our framework while the first one cannot.

It has been shown in [ZZY15] that any set of rules which ensures the finiteness of the frontier-restricted chase can be rewritten into an equivalent weakly-acyclic set of rules. This means that with respect to expressivity, all those classes are equivalent. However, this rewriting can be arbitrarily larger than the initial set of rules, and thus, studying these classes is still interesting from a succintness point of view.

3.2.1 Dependency-based Approach

The dependency-based approach aims at avoiding cyclic triggering of rules. It relies on a directed graph called the Graph of Rule Dependencies, first defined in [Bag04], and makes use of the notion of dependency between rules. Intuitively, a rule R_2 depends on a rule R_1 if there is some set of facts $\mathcal{F}$ such that an application of R_1 on $\mathcal{F}$ will lead to trigger a "new" and "useful" application of R_2.

Definition 3.10 (Rule Dependency [Bag04, GHK+13])
Given $R_1 = B_1 \to H_1$ and $R_2 = B_2 \to H_2$ two rules, R_2 depends on R_1 *if there exists some set of facts $\mathcal{F}$ such that:*

(i) there is a homomorphism π_1 from B_1 to $\mathcal{F}$,

(ii) there is a homomorphism π_2 from B_2 to $\mathcal{F}' = \alpha(\mathcal{F}, R_1, \pi_1)$,

(iii) π_2 is not a homomorphism from B_2 to $\mathcal{F}$,

(iv) $\alpha(\mathcal{F}', R_2, \pi_2) \neq \mathcal{F}'$.

In the previous definition, point (ii) and (iii) ensure that after the application of R_1 there is a new homomorphism from B_2, while point (iv) ensures this homomorphism is useful.

Example 3.7
Consider rules $R_1 = r(x_1, y_1) \wedge q(y_1) \to p(y_1, z_1)$ and $R_2 = p(x_2, y_2) \to r(y_2, z_2)$. Rule R_2 depends on rule R_1, indeed, let $F = r(a, b), q(a)$, there is a homomorphism $\pi_1 = \{(x_1, a), (y_1, b)\}$ from B_1 to F (condition (i)), there is a homomorphism $\pi_2 = \{(x_2, b), (y_2, z_1)\}$ from B_2 to $\mathcal{F}' = \alpha(\mathcal{F}, R_1, \pi_1)$ (condition (ii)), which is not a

homomorphism from B_2 to $\mathcal{F}$ (condition (iii)), and $\mathcal{F}' = \{r(a,b), q(a), p(b,z_1)\} \neq \{r(a,b), q(a), p(b,z_1), r(z_1,z_2)\} = \alpha(\mathcal{F}', R_2, \pi_2)$ (condition (iv)).

However rule R_1 does not depend on rule R_2, indeed, it can be shown that no set of facts can be built such that all conditions are verified.

While this definition is not really operational, the existence of a dependency between two rules can be computed thanks to piece-unifiers as stated by the next proposition (we do not recall the definition of atom erasing since it is not needed later).

Proposition 3.5 ([BLMS11])

Let $R_1 = B_1 \to H_1$ and $R_2 = B_2 \to H_2$ be two rules; R_2 depends on R_1 if and only if there exists an atom-erasing piece-unifier of B_2 with H_1.

Example 3.8

Consider again rules R_1 and R_2 from Example 3.7. There is a piece-unifier

$$\mu_1 = \{\{p(x_2, y_2)\}, \{p(y_1, z_1)\}, \{\{x_2, y_1\}, \{y_2, z_1\}\}\}$$

of B_2 with H_1, but there is no piece-unifier of B_1 with H_2. Indeed the unifier

$$\mu_2 = \{\{r(x_1, y_1)\}, \{r(y_2, z_2)\}, \{\{x_1, y_2\}, \{y_1, z_2\}\}\}$$

does not satisfy the piece condition of a piece-unifier since variable y_1 is unified with the existential variable z_2 and therefore $q(y_1)$ must be in the unified part of the piece-unifier.

Definition 3.11 (Graph of Rule Dependencies)

Given a set of rules $\mathcal{R}$, its graph of rule dependencies *denoted by $GRD(\mathcal{R})$ is the labelled directed graph whose set of vertices is the set of rules and where there is an edge from R_1 to R_2 if R_2 depends on R_1.*

Intuitively, this graph encodes whether a rule can lead to trigger another. As pointed out in Proposition 3.5, one can check whether a rule depends on another by finding a particular piece-unifier of the body of the latter with the head of the former.

Example 3.9 (Graph of Rule Dependencies)

Consider the rules $R_1 = p(x_1, y_1) \to q(y_1)$, $R_2 = q(x_2) \to r(x_2, y_2)$. There is a unifier $(\{q(x_2)\}, \{\{y_1, x_2\}\}, \{q(y_1)\})$ of B_2 with H_1, hence an edge from rule R_1 to R_2 but neither a unifier of B_1 with H_2, nor of B_1 with H_1 nor B_2 with H_2 ; hence, the resulting graph of rule dependencies is acyclic.

If we add the rule $R_3 = r(x_3, y_3) \to p(x_3, y_3)$, there is a unifier of B_1 with H_3 and of B_3 with H_2. Therefore, there is a cycle in the graph of rule dependencies as shown in Figure 3.1.

Figure 3.1: GRD of $\{R_1, R_2\}$ (left) and $\{R_1, R_2, R_3\}$ (right) from Example 3.9

Definition 3.12 (Acyclic Graph of Rule Dependencies)
Let $\mathcal{R}$ be a set of rules. We say that $\mathcal{R}$ is aGRD if $GRD(\mathcal{R})$ is acyclic.

Proposition 3.6 ([Bag04])

Let $\mathcal{R}$ be a set of rules. If $\mathcal{R}$ is *aGRD*, then for any set of facts $\mathcal{F}$, the oblivious chase halts on $(\mathcal{F}, \mathcal{R})$.

From a complexity point of view, it has been shown that recognising set of rules satisfying *aGRD* is a *coNP-complete* problem.

Proposition 3.7 (Complexity of *aGRD* Recognition [Bag04])

Given a set of rules $\mathcal{R}$, checking if $\mathcal{R}$ is *aGRD* is *coNP-complete*.

3.2.2 Position-based Approach

In the position-based approach, the way existential variables are "propagated" through predicate positions is analysed. The idea is to check that no generated variable appearing in some predicate position can lead to generate another one in the same position.

Various conditions based on this idea have been proposed, we will show later that all of these can be seen as acyclicity conditions. The simplest condition (and first to be defined) is that of weak-acyclicity which relies on the predicate position graph.

For the following we denote by (p, i) the i^{th} position of predicate p.

Definition 3.13 (Predicate Position Graph)
Given a set of rules $\mathcal{R}$, its predicate position graph*, denoted by $PPG(\mathcal{R})$ is the directed labelled graph whose set of vertices is the set of predicate positions of $\mathcal{R}$. Then for each rule $R \in \mathcal{R}$ and each frontier variable x in B occuring in some position (p, i), edges with origin (p, i) are built as follows: there is an edge from (p, i) to each position (q, j) in H where x occurs, and there is a* special *edge from (p, i) to each position (q, j) in H where some existential variable y appears.*

Note that in this graph, both vertices and edges are labelled.

Example 3.10 (Predicate Position Graph)
Consider rules $R_1 = p(x, y) \to s(y, z)$ *and* $R_2 = s(x, y) \to p(x, y)$. *The resulting predicate position graph is pictured on Figure 3.2.*

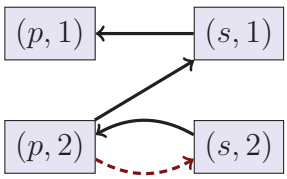

Figure 3.2: Predicate Position Graph of $\{R_1, R_2\}$ from Example 3.10

Definition 3.14 (Weak-acyclicity [FKMP05])
A set of rules $\mathcal{R}$ *is* weakly-acyclic *if* $PPG(\mathcal{R})$ *contains no cycle going through a special edge.*

Example 3.11 (Weak-acyclicity)
The set of rules $\{R_1, R_2\}$ *from Example 3.10 is not weakly-acyclic, indeed there is a cycle going through a special edge:* $((p, 2), (s, 2), (p, 2))$. *If we replace* R_2 *by* $R'_2 = s(x, y) \to p(y, x)$, *the predicate position graph of* $\{R_1, R'_2\}$ *does not contain a special cycle anymore, as can be seen on Figure 3.3; thus this set of rules is weakly-acyclic.*

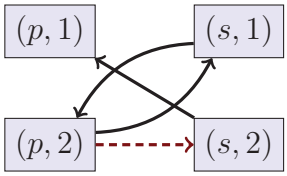

Figure 3.3: Predicate Position Graph of $\{R_1, R'_2\}$ from Example 3.11

Proposition 3.8 ([FKMP05])

Let $\mathcal{R}$ be a set of rules. If $\mathcal{R}$ is weakly-acyclic, then for any set of facts $\mathcal{F}$, σ_{fr}-$chase(\mathcal{F}, \mathcal{R})$ is finite.

Following this idea, several extensions to weak-acyclicity have been proposed, by shifting the focus either from positions to existential variables (*ja*, [KR11]), or to

positions in atoms instead of predicates (*swa*, [Mar09]). Other related notions can be imported from logic programming, e.g., finite domain [CCIL08].

We recall here their definitions before unifying them in Section 3.3 and extending them in Section 3.4.

Given a set of atoms A, we denote by $Pos_A(x)$ the set of predicate positions where the variable x occurs in A.

We first consider the notion of finite domain [CCIL08]. It relies on the fact that if all positions are only "populated" by a finite number of individuals, then there is a finite universal model of the knowledge base; indeed, by considering only rule applications that are different with respect to the frontier of the rule, there can be only finitely many rule applications. Here we use the definition from [GHK+13], which has been shown to be equivalent to the original definition, since it uses a vocabulary closer to ours.

Definition 3.15 (Finite Domain Positions)
Let $\mathcal{R}$ be a set of rules. A predicate position (p, i) is $\mathcal{R}$-recursive *with a position (q, j) if the predicate position graph $PPG(\mathcal{R})$ contains a cycle going through (p, i) and (q, j). The set $Pos_{fd}(\mathcal{R})$ of* finite domain positions *of $\mathcal{R}$ is the largest set of positions in $\mathcal{R}$ such that: for each position $(p, i) \in Pos_{fd}(\mathcal{R})$, each rule $R = (B, H) \in \mathcal{R}$, and each head atom of H of the form $a(\vec{t})$, the following conditions hold:*

- *if the i^{th} component of $\vec{t}$ is a variable $y \in fr(R)$, then $Pos_B(y) \cap Pos_{fd}(\mathcal{R}) \neq \emptyset$*
- *if the i^{th} component of $\vec{t}$ is an existential variable of R, then, for each variable $y \in fr(R)$, some position $(q, j) \in Pos_B(y) \cap Pos_{fd}(\mathcal{R})$ exists that is not $\mathcal{R}$-recursive with (p, i).*

Then, it remains to check whether all positions in a set of rules are of finite domain.

Definition 3.16 (Finite Domain [CCIL08])
A set of rules $\mathcal{R}$ is of finite domain *(fd), if $Pos_{fd}(\mathcal{R})$ coincides with the set of all predicate positions in $\mathcal{R}$.*

This notion captures the notion of weak-acyclicity, since instead of just checking that no cycle occurs in the predicate position graph, this condition checks that even if there is a cycle for an existential variable, there may be some frontier variable (therefore used by the rule application) that does not occur in some cycle (and thus can be used only finitely many times).

Example 3.12 (Finite Domain)
Let $R_1 = p(x, y) \wedge q(y) \rightarrow r(y, z)$ and $R_2 = r(y_1, y_2) \rightarrow p(y_1, y_2)$ be two rules. The set of rules $\mathcal{R} = \{R_1, R_2\}$ is of finite domain. Indeed for each predicate position, the first point in Definition 3.15 is easily verified since no predicate occurring in a rule head also appears in the same rule body. Then the only predicate position

where an existential variable occurs is $(r, 2)$ in rule R_1, and the only frontier variable is y, it remains to check that y occurs at some predicate position which is not $\mathcal{R}$-recursive with $(r, 2)$. As can be observed on Figure 3.4, $(q, 1)$ is not $\mathcal{R}$-recursive with $(r, 2)$. Therefore, the second point of the definition is also satisfied.

It should be noted that $\mathcal{R}$ is not weakly-acyclic as there is a cycle $((p, 2), (r, 2), (p, 2))$ that goes through a special edge.

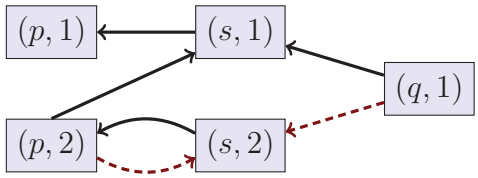

Figure 3.4: $PPG(\mathcal{R})$ from Example 3.12

Proposition 3.9 ([CCIL08])

Let $\mathcal{R}$ be a set of rules. If $\mathcal{R}$ is of finite domain, then for any set of facts $\mathcal{F}$, σ_{fr}-$chase(\mathcal{F}, \mathcal{R})$ is finite.

On the other hand, weak-acyclicity has also been generalised by shifting the focus to existential variables (instead of predicate positions). Indeed, only existential variables do matter when we consider finiteness. It makes use of another graph, that shows which existential variable can lead to create another one; then by checking that there is no cycle in this graph, we ensure that no variable can create another, and thus, that we only need finitely many fresh variables.

Definition 3.17 (Joint-acyclicity Graph)
Given a set of rules $\mathcal{R}$, and an existential variable z occuring in some rule $R = (B, H)$ of $\mathcal{R}$, let $Move(z)$ be the smallest set of predicate positions such that:

- $Pos_H(z) \subseteq Move(z)$;
- *for each rule $R' = (B', H')$ and each frontier variable y occuring in R', if $Pos_{B'}(y) \subseteq Move(z)$, then $Pos_{H'}(y) \subseteq Move(z)$.*

The joint-acyclicity graph *of $\mathcal{R}$, denoted by $JA(\mathcal{R})$ is the directed graph whose set of vertices is the set of all existential variables in $\mathcal{R}$, and where there is an edge from z_1 to z_2 whenever the rule $R = (B, H)$ that contains z_2 also contains a frontier variable y such that $Pos_B(y) \subseteq Move(z_1)$.*

Definition 3.18 (Joint-acyclicity [KR11])
A set of rules $\mathcal{R}$ is jointly-acyclic *if $JA(\mathcal{R})$ does not contain a cycle.*

Example 3.13 (Joint-acyclicity)
Let $\mathcal{R}$ be the set composed of the following four rules:

- $R_1 = p(x, y) \rightarrow r(y, z_1)$,
- $R_2 = p(x, y) \rightarrow r(z_2, y)$,
- $R_3 = r(y_1, y_2) \rightarrow s(y_1, y_2)$,
- $R_4 = s(y_1, y_2) \wedge s(y_2, y_1) \rightarrow p(y_1, y_2)$.

Then one can check that $Move(z_1) = \{(r, 2), (s, 2)\}$ and $Move(z_2) = \{(r, 1), (s, 1)\}$, and thus that $JA(\mathcal{R})$ does not contain any cycle (therefore $\mathcal{R}$ is jointly-acyclic).

It can be observed that $\mathcal{R}$ is not of finite-domain. Indeed, observe that all predicate positions but $[p, 1]$ are $\mathcal{R}$-recursive with each position but $[p, 1]$. Furthermore, in rule R_1 for instance, position $[p, 2]$ is the only position in B_1 in which the variable y occurs, but it is $\mathcal{R}$-recursive with $[r, 2]$ which holds an existential variable. Thus, position $[r, 2]$ cannot belong to the set $Pos_{fd}(\mathcal{R})$ and therefore $\mathcal{R}$ is not of finite-domain.

Proposition 3.10 ([KR11])

Let $\mathcal{R}$ be a set of rules. If $\mathcal{R}$ is jointly-acyclic, then for any set of facts $\mathcal{F}$, σ_{fr}-$chase(\mathcal{F}, \mathcal{R})$ is finite.

To define the notion of super weak-acyclicity (that shifts the focus to positions in atoms) we make use of the following notations: $[a, i]$ denotes the i^{th} position of atom a, and if p is an atom position, $term(p)$ denotes its term, and $pred(p)$ denotes its predicate. Super weak-acyclicity relies on yet another graph, and this graph uses the notion of covering between atom positions sets as defined below.

Definition 3.19 (Atom positions covering)
Given two sets of atom positions A_1 and A_2, we say that A_1 covers *A_2 if for each atom position $[p_2, i_2] \in A_2$, there is a position $[p_1, i_1] \in A_1$ and there exist two substitutions σ_1 and σ_2 such that $\sigma_1(p_1) = \sigma_2(p_2)$ and $i_1 = i_2$.*

Definition 3.20 (Super Weak-acyclicity Graph)
Given a set of rules $\mathcal{R}$, and a variable x occuring in some rule R, we define the following three sets:

- *$In(x)$ contains each atom position $[p, i]$ such that $p \in B$ and $x = term([p, i])$;*

- *$Out(x)$ contains each atom position $[p, i]$ such that $p \in H$ and $x = term([p, i])$;*
- *$Move(x)$ is the smallest set of atom positions such that:*
  - *$Out(x) \subseteq Move(x)$;*
  - *for each variable x' that is universally quantified in some rule in $\mathcal{R}$, if $Move(x)$ covers $In(x')$, then $Out(x') \subseteq Move(x)$.*

The super weak-acyclicity graph *of $\mathcal{R}$, denoted by $SWA(\mathcal{R})$ is the directed graph whose set of vertices is the set of rules, and in which there is an edge from a rule R to a rule R' if there exist a frontier variable y' from R' and an existential variable z from R, such that $Move(z)$ covers $In(y')$.*

Once the super weak acyclicity graph is built, it remains to check for its acyclicity.

Definition 3.21 (Super Weak-acyclicity [Mar09])
A set of rules $\mathcal{R}$ is super weakly-acyclic *(swa), if $SWA(\mathcal{R})$ does not contain any cycle.*

Example 3.14 (Super Weak-acyclicity)
Let $R_1 = q(y_1) \rightarrow p(y_1, z_1) \wedge p(z_1, y_1) \wedge p(y_1, y_1)$, $R_2 = p(y_2, y_2) \rightarrow s(y_2)$, and $R_3 = s(y_3) \rightarrow q(y_3)$ be rules. It can be checked that the set of rules $\mathcal{R} = \{R_1, R_2, R_3\}$ is super weakly-acyclic.

Proposition 3.11 ([Mar09])

Let $\mathcal{R}$ be a set of rules. If $\mathcal{R}$ is super weakly-acyclic, then for any set of facts $\mathcal{F}$, σ_{fr}-$chase(\mathcal{F}, \mathcal{R})$ is finite.

3.2.3 First combination

It appears that *aGRD* and *weak-acyclicity* are incomparable, as shown by the two next examples.

Example 3.15 ($wa \not\subseteq aGRD$)
Consider rules $R_1 = p(x_1, y_1) \rightarrow q(x_1, y_1)$ and $R_2 = q(x_2, y_2) \rightarrow p(y_2, x_2)$. There is a piece-unifier:

$$\mu_1 = \{\{p(x_1, y_1)\}, \{p(y_2, x_2)\}, \{\{x_1, y_2\}, \{y_1, x_2\}\}\}$$

of B_1 with H_2, therefore R_1 depends on R_2, and a piece-unifier:

$$\mu_2 = \{\{q(x_2, y_2)\}, \{q(x_1, y_1)\}, \{\{x_2, x_1\}, \{y_2, y_1\}\}\}$$

of B_2 with H_1, thus R_2 depends on R_1. It follows that $\mathcal{R} = \{R_1, R_2\}$ is not aGRD.

However, since no existential variable occurs in the rules, $\mathcal{R}$ is trivially weakly-acyclic.

Thus, the set of rules $\mathcal{R}$ satisfies wa, but not aGRD.

Example 3.16 (*$aGRD \not\subseteq wa$*)
Consider the single rule $R = p(x, y) \wedge q(y) \rightarrow p(y, z)$. There is no unifier of B with H, thus this rule does not depend on itself, inducing an acyclic graph of rule dependencies. However its predicate position graph contains a special edge loop on position $(p, 2)$, and thus is cyclic.

Therefore, $\mathcal{R} = \{R\}$ satisfies aGRD but not wa.

Therefore, a first attempt to combine both approaches has been made in [BLMS11, GHK+13]. The proposed idea was to check for position-based acyclicity conditions on each strongly connected component of the GRD. In a way, this was a "modular" approach which led to new sufficient conditions for chase termination, as states the next proposition.

Proposition 3.12 ([BLMS09, BLMS11])

Let $\mathcal{R}$ be a set of rules, and σ some derivation filter. If for any set of facts $\mathcal{F}$, the *σ-chase* of $\mathcal{F}$ w.r.t. each strongly connected component of $GRD(\mathcal{R})$ is finite, then for any set of facts $\mathcal{F}$, $\sigma\text{-}chase(\mathcal{F}, \mathcal{R})$ is finite.

In the next section, we devise a tool that combines these two approaches in a more powerful way, allowing us to not only express all these acyclicity properties in a unified way, but also to extend them.

3.3 Unifying both Approaches

In this section, we propose a new graph that encodes both atom positions and rule dependencies.

We recall that we write $[a, i]$ to denote the i^{th} position of atom a, and if p is an atom position, $term(p)$ to denote its term, and $pred(p)$ to denote its predicate. We also say that $[a, i]$ is an *existential position* (resp. *frontier position*) if $term([a, i])$ is an existential (resp. frontier) variable.

Definition 3.22 (Basic Position Graph)
Given a rule R, the individual position graph of R, denoted by $PG(R)$, is the directed graph whose vertices are the atom positions in R and there is an edge from position $[a, i]$ to $[b, j]$ if $term([a, i])$ is a frontier variable x and $term([b, j])$ is either x or some existential variable. Given a set of rules $\mathcal{R}$, the basic position graph *of $\mathcal{R}$, denoted by $PG(\mathcal{R})$, is the disjoint union of $PG(R_i)$ for all $R_i \in \mathcal{R}$.*

Example 3.17 (Basic Position Graph)
Consider the rules $R_1 = q(x_1) \rightarrow p(x_1, y_1)$ *and* $R_2 = p(x_2, y_2), s(y_2) \rightarrow q(y_2)$*, the basic position graph of* $\{R_1, R_2\}$ *is depicted in Figure 3.5 (existential positions are colored in red).*

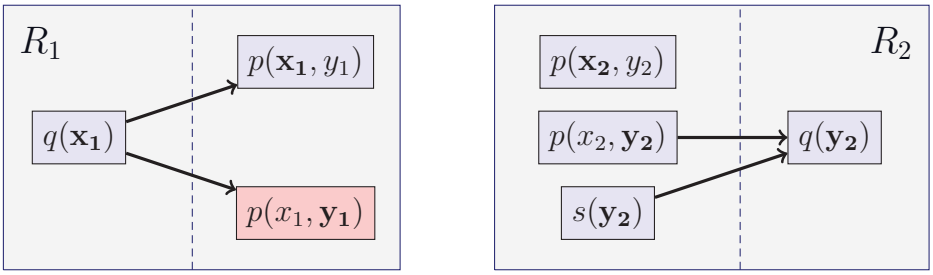

Figure 3.5: Basic Position Graph of $\{R_1, R_2\}$ from Example 3.17

An existential position $[a, i]$ is said to be *infinite* if there is a set of atoms $\mathcal{F}$, such that running the chase on $\mathcal{F}$ produces an unbounded number of instantiations of $term([a, i])$. Since we aim at detecting these infinite existential positions, we encode how variables may be "propagated" among rules by adding edges to $PG(\mathcal{R})$, called *transition edges*, which go from vertices corresponding to atom positions occurring in rule heads to vertices corresponding to atom positions in rule bodies. If X is a set of transition edges, we denote by $PG^X(\mathcal{R})$ the position graph $PG(\mathcal{R})$ in which are added the edges of X. Furthermore, we say that a set of transition edges is *correct* if it satisfies the following condition: if an existential position $[a, i]$ is infinite, then there a cycle going through $[a, i]$ in the resulting graph.

To define more formally the notion of correct sets of transition edges, we make use of the support graph of a derivation.

Definition 3.23 (Supports)
Let $\mathcal{F}$ *be a set of facts,* $\mathcal{R}$ *be a set of rules and* $D = (\emptyset, \emptyset, F_0 = \mathcal{F}), \ldots, (R_k, \pi_k, F_k)$ *be a (finite) breadth-first derivation. Let* h *be an atom occuring in the head of some rule* R_i *and* b *be an atom occuring in the body of some rule* R_j*. We say that* (h, π_i) *is a* support *for* (b, π_j) *in* D *if* $\pi_i^{safe}(h) = \pi_j(b)$*. If there is an atom* $f \in F_0 = \mathcal{F}$ *such that* $f = \pi_j(b)$*, we also say that* $(f, init)$ *is a* support *of* (b, π_j)*. Among all possible supports for* (b, π_j)*, its* first supports *are the supports* (h, π_i) *such that either* i *is minimal or* $\pi_i = init$*. By extension we say that* (R_i, π_i) *is a* support *for* (R_j, π_j) *in* D *when there exist an atom* $h \in R_i$ *and an atom* $b \in R_j$ *such that* (h, π_i) *is a support for* (b, π_j) *in* D*. Among all possible supports for* (R_j, π_j) *its* last support *is the support* (R_i, π_i) *such that* i *is maximal.*

Definition 3.24 (Support Graph)
Given a set of facts $\mathcal{F}$*, a set of rules* $\mathcal{R}$ *and a breadth-first derivation* $D = (\emptyset, \emptyset, F_0 =$

$\mathcal{F}), \ldots, (R_k, \pi_k, F_k)$, *the* support graph *of D is the directed graph with $k+1$ vertices: F_0 and the (R_i, π_i). There is an edge from a vertex v_i to $v_j = (R_j, \pi_j)$ if v_i is a support of v_j. Such an edge is called* last support edge *(LS edge) when v_i is a last support of v_j. It is called* non-transitive edge *(NT edge), if it is not a transitivity edge. A path in which all edges are either LS or NT is called a* triggering path*.*

We now define the notion of triggering derivation sequence from an atom h to an atom b as follows.

Definition 3.25 (Triggering Derivation Sequence)
Given a knowledge base $(\mathcal{F}, \mathcal{R})$ and two atoms h and b, a triggering derivation sequence *from h to b, denoted as $h \to b$ triggering sequence, is a (finite) breadth-first derivation from $\mathcal{F}$ to F_k such that (h, π_1) is a first support of (b, π_k).*

We are now able to define formally the notion of correct set of transition edges.

Definition 3.26 (Correct Set of Transition Edges)
Let $\mathcal{R}$ be a set of rules. A set of transition edges X is correct *if, whenever there exists an $h \to b$ triggering derivation sequence, $PG^X(\mathcal{R})$ contains a transition from (h, i) to (b, i) for all $1 \leq i \leq r$ where r is the arity of the predicate of h (and b).*

Theorem 3.2

Let $\mathcal{R}$ be a set of rules, and X a set of transition edges. If X is correct, and $PG^X(\mathcal{R})$ is acyclic, then for any set of facts $\mathcal{F}$, σ_{fr}-*chase*$(\mathcal{F}, \mathcal{R})$ is finite.

Proof: Observe that if there is no cycle in a correct PG^X, then no existential variable generated by a rule R can be "used" to generate a fresh existential variable in the same position by another application of R. Thus, only a finite number of fresh existential variables is needed in σ_{fr}-*chase*$(\mathcal{F}, \mathcal{R})$, and therefore, σ_{fr}-*chase*$(\mathcal{F}, \mathcal{R})$ is finite. □

In the following we propose different position graphs that have correct sets of transition edges. First, the full position graph corresponds to the case where all rules are supposed to depend on all rules. It is used to check for position-based acyclicity conditions without taking into account the "true" dependencies between rules.

Definition 3.27 (Full Position Graph)
Given a set of rules $\mathcal{R}$, the full position graph *of $\mathcal{R}$, denoted by $PG^F(\mathcal{R})$, is obtained from $PG(\mathcal{R})$ by adding a transition edge from each position $[h, k]$ in a rule head H_i to each position $[b, k]$ in a rule body B_j with the same predicate.*

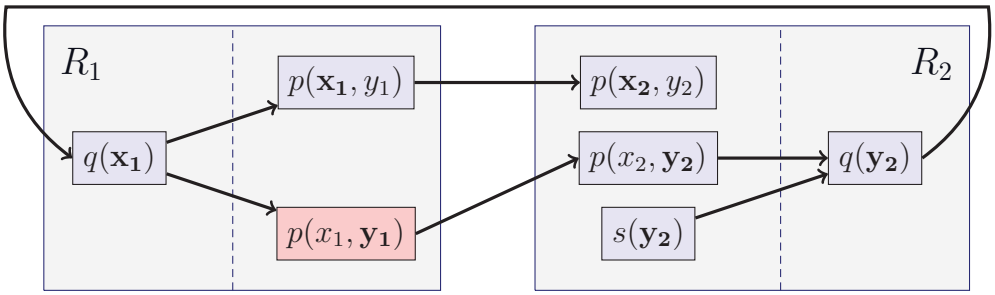

Figure 3.6: Full Position Graph of $\{R_1, R_2\}$ from Example 3.17

Example 3.18 (Full Position Graph)
Consider rules $\{R_1, R_2\}$ from Example 3.17. The full position graph of $\{R_1, R_2\}$ is represented on Figure 3.6.

Proposition 3.13

The set of transition edges of $PG^F(\mathcal{R})$ is correct for any set of rules $\mathcal{R}$.

Proof: Follows immediately from the definitions. □

The full position graph is sufficient to encode all position-based acyclicity conditions defined in Section 3.2.2. To this end, we define marking functions, that intuitively assign to each atom position a subset of positions reachable from it, according to some propagation constraints depending on the acyclicity property we want to express. Then, we say that the property is fulfilled when no existential position can be reached from itself.

Definition 3.28 (Marking Function)
A marking function *Y assigns to each vertex $[a, i]$ in some position graph PG^X a subset of its (direct or indirect) successors, called its* marking.

Definition 3.29 (Marked Cycle)
Given a set of rules $\mathcal{R}$, a marking function Y, a set of transition edges X, and an atom position $[a, i]$ occuring in $\mathcal{R}$, a marked cycle *for $[a, i]$ w.r.t. X, Y and $\mathcal{R}$ is a cycle C in $PG^X(\mathcal{R})$ such that $[a, i] \in C$ and for all $[a', i'] \in C$, $[a', i']$ belongs to the marking of $[a, i]$.*

Obviously, the less situations there are in which the marking may "propagate" in a position graph, the stronger the acyclicity property is.

Definition 3.30 (Acyclicity Property)
Given a marking function Y and a set of transition edges X, the acyclicity property *associated with Y in PG^X, denoted by Y^X, is satisfied by a set of rules $\mathcal{R}$ if there is no marked cycle for an existential position in $PG^X(\mathcal{R})$.*

Theorem 3.3

Let $\mathcal{R}$ be a set of rules and Y be an acyclicity property such that if $\mathcal{R}$ satisfies Y then for any set of facts $\mathcal{F}$ σ_{fr}-*chase*$(\mathcal{F}, \mathcal{R})$ is finite, and X be a correct set of transition edges. If $\mathcal{R}$ satisfies Y^X, then for any set of facts $\mathcal{F}$, σ_{fr}-*chase*$(\mathcal{F}, \mathcal{R})$ is finite.

Proof: Let us say that a transition edge from $[a, i]$ in R_1 to $[a', i]$ in R_2 is *useful* if there are a set of facts F and a homomorphism π_1 from B_1 to F such that there is a homomorphism π_2 from B_2 to $F' = \alpha(F, R_1, \pi_1)$ and $\pi_1^{safe}(a) = \pi_2(a')$. Furthermore we say that the application of R_2 *uses* edge $([a, i], [a', i])$.

One can see that a useful edge exactly corresponds to an $h \to b$ triggering derivation sequence where $[a, i]$ occurs in h and $[a', i]$ occurs in b. It follows from the correctness of X that no useful edge is removed.

Now let Y be an acyclicity property ensuring the finiteness of the frontier-restricted chase. Assume there is a a set of rules $\mathcal{R}$ that satisfies Y^X and there is a set of facts $\mathcal{F}$ such that σ_{fr}-*chase*$(\mathcal{F}, \mathcal{R})$ is infinite. Then there is a rule application in this (infinite) derivation that uses a transition edge $([a, i], [a', i])$ which does not appear in $PG^X(\mathcal{R})$. This edge is then useful, and thus $PG^X(\mathcal{R})$ does not use a correct set of edges. □

We now define marking functions corresponding to acyclicity conditions defined in Section 3.2.

We first define three properties of a marking $M([a, i])$ that may be fulfilled by the different marking functions, which allows us to easily compare them:

- $(\mathbf{P_1})$ $\Gamma([a, i]) \subseteq M([a, i])$;
- $(\mathbf{P_2})$ for all $[a', i'] \in M([a, i])$ such that $[a', i']$ occurs in some rule head: $\Gamma([a', i']) \subseteq M([a, i])$;
- $(\mathbf{P_3})$ for all variable v in a rule body, such that for all positions $[a', i']$ with $term([a', i']) = v$, there is a position $[a'', i'] \in M([a, i])$ with $pred([a', i']) = pred([a'', i'])$ and $term([a'', i']) = v$: $\Gamma(v) \subseteq M([a, i])$, where $\Gamma(v)$ is the union of all $\Gamma(p)$ where p is an atom position in which v occurs.

Property $(\mathbf{P_1})$ states that the neighbourhood of the position we are considering must be in the marking; then property $(\mathbf{P_2})$ that if a position belonging to the

marking occurs in some rule head, then its neighbourhood must also belong to it; finally property ($\mathbf{P_3}$) states that if for all positions in which some variable v occurs in a rule body, there is a position belonging to the marking with the same predicate position in which v occurs, then the neighbourhood of variable v (that is all neighbours of all positions in which v occurs) must also belong to the marking.

The first position-based acyclicity condition is weak-acyclicity (see Definition 3.14 for its original wording).

Definition 3.31 (Weakly-acyclic Marking)
A marking M is a weakly-acyclic marking *with respect to some transition edge condition X, if for any set of rules $\mathcal{R}$ and any position $[a, i] \in PG^X(\mathcal{R})$, $M([a, i])$ is the minimal set such that:*

- ($\mathbf{P_1}$) *holds,*
- $\forall [a', i'] \in M([a, i]), \Gamma([a', i']) \subseteq M([a, i])$.

Observe that the latter condition implies ($\mathbf{P_2}$) and ($\mathbf{P_3}$).

Example 3.19
Consider the set of rules $\mathcal{R}$ composed of rule $R_1 = p(x_1, y_1) \to s(y_1, z_1)$ and rule $R_2 = s(x_2, y_2) \to p(x_2, y_2)$ from Example 3.10 whose predicate position graph is pictured on Figure 3.2.

The full position graph $PG^F(\mathcal{R})$ is depicted by Figure 3.7.

There is a single existential position $[s(y_1, z_1), 2]$ (from rule R_1) whose marking M is the following: first M contains all neighbours of $[s(y_1, z_1), 2]$, that is $[s(x_2, y_2), 2]$ from rule R_2. Then, the second condition of the weakly-acyclic marking forces all neighbours of each element of the marking to also be in the marking. Therefore $[p(x_2, y_2), 2]$ also belongs to M, and for the same reason $[p(x_1, y_1), 2]$ and $[s(y_1, z_1), 2]$ Hence, we have:

$$M([s(y_1, z_1), 2]) = \{[s(x_2, y_2), 2], [p(x_2, y_2), 2], [p(x_1, y_1), 2], [s(y_1, z_1), 2]\}$$

and since M contains the existential variable from which we build it, $\mathcal{R}$ is not weakly-acyclic.

The following lemma will be of great use in the different proofs of equivalence between the original acyclicity conditions and their markings in the full position graph. It exhibits the link between the edges in the predicate position graph of a set of rules, and those in its full position graph.

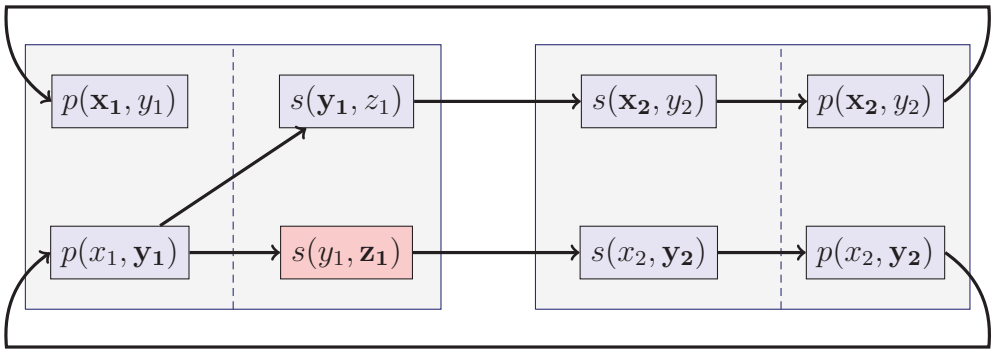

Figure 3.7: $PG^F(\mathcal{R})$ from Example 3.19

Lemma 3.1

Let $\mathcal{R}$ be a set of rules. For each edge $((p,i),(q,j))$ in the predicate position graph of $\mathcal{R}$, there is the following non-empty set of edges in $PG^F(\mathcal{R})$:

$$E_{(p,i),(q,i)} = \{([a,i],[b,j]) \mid pred([a,i]) = p \text{ and } pred([b,j]) = q\}$$

Furthermore, the set of all $E_{(p,i),(q,j)}$ for all positions (p,i) and (q,j) forms a partition of all edges in $PG^F(\mathcal{R})$.

Proof: Follows immediately from the construction of PG^F. □

Proposition 3.14

A set of rules $\mathcal{R}$ is *wa* iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the *wa* marking (formally defined in Definition 3.31).

Proof: If $\mathcal{R}$ is not *wa*, then there is some cycle in the graph of predicate positions going through a special edge. Let (p,i) be the predicate position where this edge ends, and z be an existential variable which occurs in (p,i). Let M be the wa-marking of any existential position $[a,i]$ with $pred([a,i]) = p$ and $term([a,i]) = z$. Condition $(\mathbf{P_1})$ ensures that the successors of $[a,i]$ are marked; then the propagation can be seen as performing a classic breadth-first traversal of the graph. By Lemma 3.1, to each cycle in the graph of predicate positions of $\mathcal{R}$ corresponds a set of cycles in $PG^F(\mathcal{R})$. Since (p,i) belongs to a cycle, $[a,i]$ is obviously marked by the propagation. Hence $PG^F(\mathcal{R})$ does not satisfy the acyclicity property associated with the *wa*-marking.

Conversely, if $\mathcal{R}$ is *wa*, there is no cycle going through a special edge in $PPG(\mathcal{R})$. By Lemma 3.1, no cycle in $PG^F(\mathcal{R})$ goes through an existential position; therefore $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the *wa*-marking. □

Definition 3.32 (Finite Domain Marking)
A marking M is a finite domain marking *with respect to some transition edge condition X, if for any set of rules $\mathcal{R}$ and any position $[a,i] \in PG^X(\mathcal{R})$, $M([a,i])$ is the minimal set such that:*

- $(\mathbf{P_1})$ *and* $(\mathbf{P_3})$ *hold,*
- $\forall [a',i'] \in M([a,i]), \Gamma([a',i']) \setminus \{[a,i]\} \subseteq M([a,i])$.

It should be pointed out that the latter condition implies $(\mathbf{P_2})$.

Proposition 3.15

A set of rules $\mathcal{R}$ is *fd* iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the *fd* marking (Definition 3.32).

Proof: Let $\mathcal{R}$ be a set of rules that is *fd*. Then, for each existential position $[p,i]$ there exists a position $[p,j]$ for each variable of the frontier in the predicate positions graph such $[p,j]$ does not belong to a cycle. Given $PG^F(\mathcal{R})$, we can see that Condition $(\mathbf{P_3})$ ensures that $\mathcal{R}$ is *fd*. □

Definition 3.33 (Joint-acyclicity Marking)
A marking M is a joint-acyclicity marking *with respect to some transition edge condition X, if for any set of rules $\mathcal{R}$ and any position $[a,i] \in PG^X(\mathcal{R})$, $M([a,i])$ is the minimal set such that:* $(\mathbf{P_1})$, $(\mathbf{P_2})$ *and* $(\mathbf{P_3})$ *hold.*

Proposition 3.16

A set of rules $\mathcal{R}$ is *ja* iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the *ja* marking (Definition 3.33).

Proof: Note that the definition of the *ja* marking is defined in the same way as the Move set from Definition 3.17. Furthermore, by Lemma 3.1, for any predicate position (p,i) in the JA graph of a set of rules $\mathcal{R}$, there is a cycle going through (p,i) iff for any existential position $[a,i]$ such that $pred([a,i]) = p$ we have $[a,i] \in M([a,i])$. □

Definition 3.34 (Super-weakly-acyclic Marking)
A marking M is a super-weakly-acyclic marking *with respect to some transition edge condition X, if for any set of rules $\mathcal{R}$ and any position $[a,i] \in PG^X(\mathcal{R})$, $M([a,i])$ is the minimal set such that:*

- $(\mathbf{P_1})$ *and* $(\mathbf{P_3})$ *hold,*
- *for all* $[a',i'] \in M([a,i])$ *occuring in a rule head,* $\{[a'',i'] \in \Gamma([a',i']) \mid a'$ *and* a'' *unify*$\} \subseteq M([a,i])$.

Proposition 3.17

A set of rules $\mathcal{R}$ is *swa* iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the *swa* marking (Definition 3.34).

Proof: Similarily to the proof of Proposition 3.16, observe that the swa marking is defined in the same way as the move set of Definition 3.20. □

We synthesise here propositions 3.14 to 3.17.

Theorem 3.4

A set of rules $\mathcal{R}$ is *wa* (resp. *fd*, *ja*, *swa*) iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the *wa* (resp. *fd*, *ja*, *swa*) marking.

Proof: Follows immediately from Propositions 3.14 to 3.17. □

As already mentioned in Section 3.2, all acyclicity properties can be safely combined with the GRD (initially proposed in [BLMS11]). More formally, we recall the definition from [GHK+13] in which the previous definition is slightly improved:

Definition 3.35 ($Y^\prec$ [BLMS11, GHK+13])
Given an acyclicity property Y, a set of rules $\mathcal{R}$ satisfies $Y^\prec$ if each strongly connected component of $GRD(\mathcal{R})$ satisfies Y, except for those composed of a single rule and no loop.

The "no loop" condition is here to ensure that if $GRD(\mathcal{R})$ is acyclic, then $\mathcal{R}$ satisfies $Y^\prec$ for any Y.

To capture these conditions, we define a second transition edge condition for position graphs.

Definition 3.36 (Dependency Position Graph)
Given a set of rules $\mathcal{R}$, the dependency position graph *of $\mathcal{R}$, denoted by $PG^D(\mathcal{R})$, is obtained from $PG(\mathcal{R})$ by adding a transition edge from each position $[h, k]$ in a rule head H_i to each position $[b, k]$ in a rule body B_j with the same predicate if there is a path from R_i to R_j in $GRD(\mathcal{R})$, i.e., if R_j depends directly or indirectly on R_i.*

Example 3.20 (Dependency Position Graph)
Let $R_1 = r(x_1, y_1) \to p(z_1, y_1) \wedge q(y_1)$ and $R_2 = p(x_2, y_2) \wedge q(x_2) \to p(y_2, z_2)$ be two rules. The dependency position graph of $\mathcal{R} = \{R_1, R_2\}$ is depicted on Figure 3.8. Note that there is a piece-unifier of B_2 with H_1: $\mu_1 = \{\{q(x_2)\}, \{\{x_2, y_1\}\}, \{q(y_1)\}\}$ and a piece-unifier of B_1 with H_2: $\mu_2 = \{\{r(x_1, y_1)\}, \{\{x_1, y_2\}, \{y_1, z_2\}\}, \{p(y_2, z_2\}\}$.

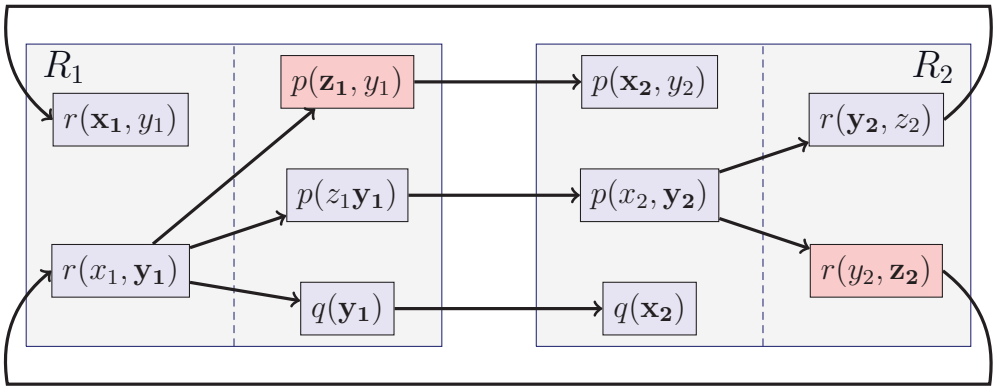

Figure 3.8: $PG^D(\mathcal{R})$ from Example 3.20

The first thing to notice is that for any set of rules the dependency position graph is a subgraph of the full position graph.

Proposition 3.18

Let $\mathcal{R}$ be a set of rules, $E(PG^D(\mathcal{R})) \subseteq E(PG^F(\mathcal{R}))$.

Proof: Follows immediately from the definitions. □

However, the correctness of its set of edges still holds.

Proposition 3.19

The set of transition edges of $PG^D(\mathcal{R})$ is correct for any set of rules $\mathcal{R}$.

The proof of this proposition relies on the following two lemmas.

Lemma 3.2

If D is an $h \to b$ triggering derivation sequence of length k, then there is a triggering path from (R_1, π_1) to (R_k, π_k) in the support graph of D.

Proof: There is an edge from (R_1, π_1) to (R_k, π_k) in the support graph of D. By removing transitivity edges, it remains a path from (R_1, π_1) to (R_k, π_k) for which all edges are either LS or NT. □

Lemma 3.3

If there is an edge from (R_i, π_i) to (R_j, π_j) that is either LS or NT in the support graph of D, then R_j depends on R_i.

Proof: Assume there is a LS edge from (R_i, π_i) to (R_j, π_j) in the support graph of D. Then the applicatin of R_i according to π_i on F_{i-1} produces F_i on which all atoms required to map B_j are present (or it would not have been a last support). Since it is a support there is also an atom required to map B_j that appeared on F_{i-1}. It follows that R_j depends on R_i.

Suppose now that the edge is NT. Consider F_ℓ such that there is a LS edge from (R_ℓ, π_ℓ) to (R_j, π_j). See that there is no path from (R_i, π_i) to (R_ℓ, π_ℓ) (otherwise there would be a path from (R_i, π_i) to (R_j, π_j) and the edge would be a transitive edge). Thus we can consider the set of facts $F_{\ell \setminus i}$ that would have been created by the following derivation sequence:

1. first apply from F_0 all rule applications of the initial sequence until (R_{i-1}, π_{i-1}),
2. then apply all possible applications of this sequence from $i+1$ until ℓ.

We can apply (R_i, π_i) on the set of facts $F_{\ell \setminus i}$ thus obtained (since it contains all atoms from F_{i-1}. Let us now consider the set of facts F' obtained after this rule application. We must now check that (R_j, π_j) can be applied on F'. This stems from the fact that there is no support path from (R_i, π_i) to (R_j, π_j). This last rule application relies upon an atom that is introduced by the application of (R_i, π_i), thus R_j depends on R_i. □

We can now conclude by the proof of Proposition 3.19:

Proof: If there is an $h \to b$ triggering derivation sequence, then by Lemma 3.2, we can exhibit a triggering path that corresponds to a path in the graph of rule dependencies (Lemma 3.3). □

The next proposition states that checking acyclicity properties in the dependency position graph is equivalent to checking acyclicity properties on each strongly connected component of the graph of rule dependencies.

Proposition 3.20

Given an acyclicity property Y and a set of rules $\mathcal{R}$, $\mathcal{R}$ satisfies $Y^{\prec}$ if and only if $\mathcal{R}$ satisfies Y^D; i.e., $Y^D = Y^{\prec}$.

Proof: Let $\mathcal{R}$ be a set of rules and Y be an acyclicity property.

We first show that if $\mathcal{R}$ is not Y^D then it is not $Y^{\prec}$. Suppose that $\mathcal{R}$ does not satisfy Y^D. We then have an existential position $[a, i]$ in $PG^D(\mathcal{R})$ such that $[a, i] \in M([a, i])$, where M is the marking associated with Y. Specifically, this means that there is a cycle going through $[a, i]$ in $PG^D(\mathcal{R})$. Then all rules from this cycle belong to the same strongly connected component of $GRD(\mathcal{R})$. Consider the restriction of $\mathcal{R}$ to the set of rules $\mathcal{R}'$ that correspond to the S.C.C. in which the rules from this cycle appear. If we build $PG^F(\mathcal{R})$, we see that $\mathcal{R}'$ does not satisfy Y^F, hence Y. We have then exhibit a S.C.C. of the $GRD(\mathcal{R})$ that does not satisfy Y, hence $\mathcal{R}$ is not $Y^{\prec}$.

Now we show that if $\mathcal{R}$ is not $Y^{\prec}$, then it is not Y^D. Assume that $\mathcal{R}$ does not satisfy $Y^{\prec}$. Since it does not satisfy $Y^{\prec}$ there is at least one S.C.C. that does not satisfy Y. Call it $\mathcal{R}'$. Hence $PG^F(\mathcal{R}')$ contains an existential position $[a, i]$ belonging to a cycle. Since $\mathcal{R}$ (hence $\mathcal{R}'$) is Y^D, this cycle does not occur anymore in $PG^D(\mathcal{R}')$. However, the only edges we are allowed to remove in $PG^D(\mathcal{R}')$ are edges between rules R_i and R_j for which there is no path from R_i to R_j in $GRD(\mathcal{R})$. Thus, we cannot remove any edge (from the definition of a S.C.C.). Hence, $\mathcal{R}'$ is not Y^D. □

From the definition of an acyclicity property that we proposed earlier (Definition 3.30), we can prove in a generic way that some properties are stronger (or weaker) than others. In the following, given two acyclicity properties Y_1 and Y_2 we say that Y_1 is *weaker* than Y_2, denoted by $Y_1 \subseteq Y_2$ if for any set of rules $\mathcal{R}$, $\mathcal{R}$ satisfies Y_1 implies that $\mathcal{R}$ satisfies Y_2. If, furthermore, $Y_2 \nsubseteq Y_1$, we write $Y_1 \subset Y_2$.

First, we can prove that if an acyclicity property does not generalise $aGRD$, the acyclicity property obtained by checking it on PG^D instead of PG^F is strictly stronger than the original one. The idea is that by switching to PG^D, we take into account dependencies between rules, which is why if the property already generalises $aGRD$ (and thus already takes dependencies into account in some way), it may not become strictly stronger.

Proposition 3.21

Let Y be an acyclicity property, if $aGRD \notin Y$, then $Y \subset Y^D$.

Proof: Let $\mathcal{R}$ be a set of rules that does not satisfy Y but satisfies $aGRD$. From the definition of $aGRD$, $GRD(\mathcal{R})$ is composed of $|\mathcal{R}|$ strongly connected components with no loop. Thanks to Proposition 3.20, $\mathcal{R}$ trivially satisfies Y^D. Therefore, $\mathcal{R}$ is a set of rules satisfying Y^D but not Y. □

The next proposition states that if two acyclicity properties are comparable, then the properties obtained by extending them to take into account dependencies also are.

Proposition 3.22

Let Y_1 and Y_2 be two acyclicity properties, if $Y_1 \subseteq Y_2$ then $Y_1^D \subseteq Y_2^D$.

Proof: Consider a set of rules $\mathcal{R}$ that satisfies Y_1^D. From Proposition 3.20, each strongly connected component of $GRD(\mathcal{R})$ satisfies Y_1. Since $Y_1 \subseteq Y_2$, each S.C.C. of $GRD(\mathcal{R})$ also satisfies Y_2, therefore $\mathcal{R}$ satisfies Y_2^D. □

Then, we can check that a strict inclusion of two acyclicity properties is preserved (under some minor constraints) while shifting to the dependency version.

Proposition 3.23

Let Y_1 and Y_2 be two acyclicity properties such that $Y_1 \subset Y_2$, $wa \subseteq Y_1$ and $Y_1 \subset Y_1^D$. Then, $Y_1^D \subset Y_2^D$.

Proof: From Proposition 3.22, we know that $Y_1^D \subseteq Y_2^D$; it remains to exhibit a set of rules that satisfies Y_2^D but not Y_1^D.

Let $\mathcal{R}$ be a set of rules that satisfies Y_2 but neither Y_1 nor $aGRD$. Rewrite $\mathcal{R}$ into $\mathcal{R}'$ by replacing each rule $R_i = (B_i, H_i)$ with a new rule $R_i' = (B_i \cup \{p(x)\}, H_i \cup \{p(x)\})$, where p is a fresh predicate and x a fresh variable. Each rule can now be unified with each rule, but the only created cycles are those which contain only atom positions $[p(x), 1]$, hence none of those cycles go through existential positions. Since $wa \subseteq Y_1$ (and so $wa \subseteq Y_2$), the added cycles do not change the behaviour of $\mathcal{R}$ with respect to properties Y_1 or Y_2. Hence $\mathcal{R}'$ satisfies Y_2 and not Y_1, and since $GRD(\mathcal{R}')$ is a complete graph, $PG^D(\mathcal{R}') = PG^F(\mathcal{R}')$. We can conclude that $\mathcal{R}'$ satisfies Y_2^D but not Y_1^D. □

In this section we have presented a new graph that allows to encode most existing acyclicity notions. Next, we make use of this new tool to extend them in a generic way.

3.4 Extensions

In this section, we present two other sets of transition edges for the position graph, and show that this allows us to extend the acyclicity notions detailed previously.

First, we need to define the notion of agglomerated rule, which allows us to encode information about unifiers in the position graph.

While technical, the intuition behind this definition is quite simple: we want the agglomerated rule to limit the number of existential variables from a rule. The idea is to relax the notion of piece-unifier; indeed, through a derivation, it may happen that the atoms needed to trigger a rule have actually been found, as illustrated by the next example.

Example 3.21
Let $R_1 = p(x_1, y_1) \rightarrow r(y_1, z_1) \wedge q(z_1)$, $R_2 = q(x_2) \rightarrow s(x_2)$ and $R_3 = r(x_3, y_3) \wedge s(y_3) \rightarrow p(x_3, y_3)$. We want to have a transition edge from $[r(y_1, z_1), 2]$ to $[r(x_3, y_3), 2]$ since the fresh variable generated after an application of rule R_1 can be used for a new application of rule R_3. It we only consider piece-unifiers of B_3 with H_1, this edge does not appear; indeed, there is no such piece-unifier. Thus, the idea is to "relax" the piece-unifiers, and to this purpose, we weaken which variables are existentially quantified by "moving" the frontier along the path (in this example, this path goes through rule R_2).

Then we build an agglomerated rule, that contains both R_1 and R_2 information about which variables could be unified later. In this case, the agglomerated rule of R_1 relatively to R_3 would be: $R_1^3 = p(x_1, y_1) \wedge fr(y_1) \wedge fr(z_1) \rightarrow r(y_1, z_1) \wedge q(z_1)$. Since z_1 is not existentially quantified anymore, there is a piece-unifier of B_3 with R_1^3.

Definition 3.37 (Agglomerated Rule)
Given $\mathcal{R}$ a set of rules and $R_i, R_j \in \mathcal{R}$, an agglomerated rule associated with (R_i, R_j) has the following form:

$$R_i^k = B_i \bigcup_{t \in T \subseteq terms(H_i)} fr(t) \rightarrow H_i$$

where fr is a new unary predicate that does not appear in $\mathcal{R}$, and the atoms $fr(t)$ are built as follows. Let $\mathcal{P}$ be a non-empty set of paths from R_i to direct predecessors of R_j in $GRD(\mathcal{R})$. Let $P = (R_1, \ldots, R_n)$ be a path in $\mathcal{P}$. One can associate a rule R^P with P by building a sequence $R_1 = R_1^P, \ldots, R_n^P$ such that $\forall 1 \leq l \leq n$, there is a piece-unifier μ_l of B_{l+1} with the head of R_l^P, where the body of R_{l+1}^p is

$B_l^P \cup \{fr(t) \mid t$ is a term of H_l^P unified in $\mu_l\}$, and the head of R_{l+1}^P is H_1. Note that for all l, $H_l^P = H_1$, however, for $l \neq 1$, R_l^P may have less existential variables than R_l due to the added atoms. The agglomerated rule R_i^j built from $\{R^P \mid P \in \mathcal{P}\}$ is $R_i^j = \bigcup_{P \in \mathcal{P}} R^P$.

Definition 3.38 (Position Graph with Unifiers)
Given a set of rules $\mathcal{R}$, the position graph with unifiers *of $\mathcal{R}$, denoted by $PG^U(\mathcal{R})$, is obtained from $PG(\mathcal{R})$ by adding a transition edge from each position $[h, k]$ in a rule head H_i to each position $[b, k]$ in a rule body B_j with the same predicate if there is a piece-unifier μ of B_j with the head of an agglomerated rule R_i^j such that $\sigma_\mu(term([b, k])) = \sigma_\mu(term([h, k]))$.*

Example 3.22 (Position Graph with Unifiers)
Remember rules $R_1 = r(x_1, y_1) \rightarrow p(z_1, y_1) \wedge q(y_1)$ and $R_2 = p(x_2, y_2) \wedge q(x_2) \rightarrow p(y_2, z_2)$ from Example 3.20. The position graph with unifiers of $\mathcal{R} = \{R_1, R_2\}$ is shown on Figure 3.9. Note that differently from its dependency position graph (see Figure 3.8), the position graph with unifiers does not contain any cycle.

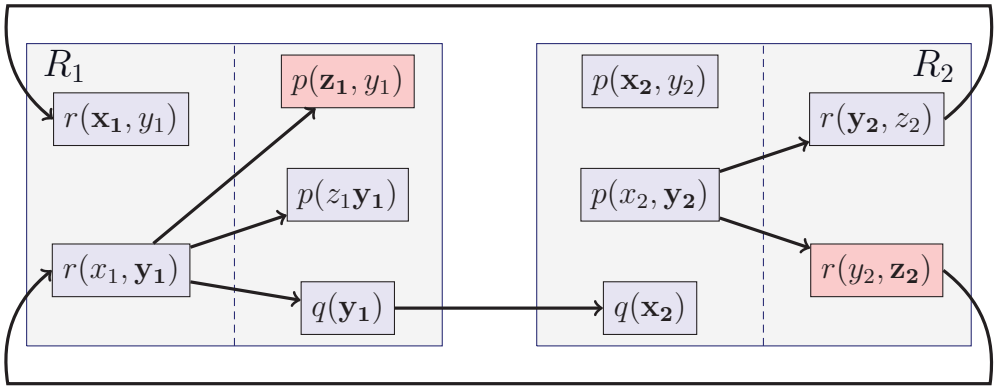

Figure 3.9: $PG^U(\mathcal{R})$ from Example 3.22

One can observe that the set of edges of the position graph with unifiers is a subset of the edges of the dependency position graph.

Proposition 3.24

Let $\mathcal{R}$ be a set of rules, $E(PG^U) \subseteq E(PG^D)$.

Proof: Follows immediately from the fact that in PG^U we only consider paths in the graph of rule dependencies to create the agglomerated rule. □

Proposition 3.25

The set of transition edges of $PG^U(\mathcal{R})$ is correct for any set of rules $\mathcal{R}$.

Proof: Consider a $h \to b$ triggering derivation sequence from $\mathcal{F}$ to F_k. We note $H^P = \pi_k(B_k) \cap \pi_1^{safe}(H_1)$ the atoms of F_k that are introduced by the application (R_1, π_1) and are used for the application (R_k, π_k). Note that this set of atoms is not empty, since it contains at least the atom produced from h. Now consider the set of terms $T^P = terms(H^P) \cap terms(\pi_k(B_k) \setminus H^P)$ that separate the atoms of H^P from the other atoms of $\pi_k(B_k)$. Now we consider the rule $R^P = B_1 \cup \{fr(t) \mid t \textit{ is a variable of } R_1 \textit{ and } \pi_1^{safe}(t) \in T^P\} \to H_1$. Consider the set of facts $F^P = F_k \setminus H^P \cup \{fr(t) \mid t \in T^P\}$. Consider the mapping π_1^P from the variables of the body of R^P to those of F^P defined as follows: it v is a variable from B_1 then $\pi_1^P(v) = \pi_1(v)$, otherwise v is a variable in an fr-atom and $\pi_1^P(v) = \pi_k(v)$. This mapping is a homomorphism, thus we can consider the set of facts $F^{P'} = \alpha(F^P, R^P, \pi_1^P)$. This application produces a new application of R_k that maps b to the atom produced from h. Indeed, consider the mapping π_k^P from the variables of B_k to those of $F^{P'}$ defined as follows: if v is a variable of B_k such that $\pi_k(v) \in terms(H^P) \setminus T^P$, then $\pi_k^P(v) = \pi_1^{P^{safe}}(v')$, where v' is the variable of H_1 that produced $\pi_k(v)$, otherwise $\pi_k^P(v) = \pi_1^{P^{safe}}(v)$. This mapping is a homomorphism. Thus, there is a piece-unifier of B_k with the head of R^P that unifies h and b.

It remains now to prove that for each atom $fr(t)$ in the body of R^P there exists a triggering path $P_i = (R'_1, \pi'_1) = (R_1, \pi_1)$ to $(R'_\ell, \pi'_\ell) = (R_k, \pi_k)$ in the support graph such that $fr(t)$ appears in the agglomerted rule R_i^A along $R_1, \ldots, R_{k-1}$. For a contradiction, let v be a variable occurring in some fr-atom in R^P. Suppose that $fr(v)$ does not appear in any agglomerated rule corresponding to a triggering path between (R_1, π_1) and (R_k, π_k). Since $\pi_1(v)$ is an existential variable generated by the application of R_1 and there is no unifier on the GRD paths that correspond to these triggering paths that unify v, $\pi_1(v)$ may only occur in atoms that are not used (even transitively) by (R_k, π_k), i.e., $\pi_1(v) \notin T^P$. There, v does not appear in an fr-atom in R^P, which leads to a contradiction. Since R^P and $R^A = \bigcup R_i^A$ have the same head and the frontier of R^P is a subset of the frontier of R^A, any unifier with R^P is also a unifier with R^A. Thus there is a unifier of R_k with R^A that unifies h and b, and there are the corresponding correct transition edges in PG^U. □

The next theorem states that the position graph with unifiers is strictly more powerful than the dependency position graph; furthermore, the "gap" from Y^D to Y^U is at least "as large" as from Y to Y^D.

Theorem 3.5

Let Y be an acyclicity property; then, $Y^D \subseteq Y^U$ and if $Y \subset Y^D$ then $Y^D \subset Y^U$. Furthermore, there is an injective mapping from the sets of rules satisfying Y^D but not Y to the sets of rules satisfying Y^U but not Y^D.

Proof: From Proposition 3.24, it follows immediately that $Y^D \subseteq Y^U$.

Now, it remains to exhibit a set of rules satisfying Y^U but not Y^D. Assume $Y \subset Y^D$ and $\mathcal{R}$ satisfies Y^D but not Y. Rewrite $\mathcal{R}$ into $\mathcal{R}'$ by applying the following steps. First for each rule $R_i = B_i[\vec{x}, \vec{y}] \to H_i[\vec{x}, \vec{y}]$, let $R_{i,1} = B_i[\vec{x}, \vec{y}] \to p_i(\vec{x}, \vec{y})$ where p_i is a fresh predicate ; and $R_{i,2} = p_i(\vec{x}, \vec{y}) \to H_i[\vec{y}][\vec{z}]$. Then for each rule $R_{i,1}$, let $R'_{i,1} = (B'_{i,1}, H_{i,1})$ such that $B'_{i,1} = B_{i,1} \cup \{p'_{j,i}(x_{j,i}) \mid \forall R_j \in \mathcal{R}\}$, where each $p'_{j,i}$ is a fresh predicate and each $x_{j,i}$ a fresh variable. Now for each rule $R_{i,2}$, let $R'_{i,2} = (B_{i,2}, H'_{i,2})$ where $H'_{i,2} = H_{i,2} \cup \{p'_{i,j}(z_{i,j}) \mid \forall R_j \in \mathcal{R}\}$, where $z_{i,j}$ are fresh existential variables. Let $\mathcal{R}' = \bigcup_{R_i \in \mathcal{R}} \{R'_{i,1}, R'_{i,2}\}$. This construction ensures that each $R'_{i,2}$ depends on $R'_{i,1}$, and each $R'_{i,1}$ depends on each $R'_{j,2}$; thus there is a transition edge from each $R'_{i,1}$ to $R'_{i,2}$ and from each $R'_{j,2}$ to each $R'_{i,1}$ in $PG^D(\mathcal{R}')$. Hence, $PG^D(\mathcal{R}')$ contains exactly one cycle for each cycle in $PG^F(\mathcal{R})$. Furthermore, $PG^D(\mathcal{R}')$ contains at least one marked cycle w.r.t. Y and then $\mathcal{R}'$ is not Y^D. Now each cycle in $PG^U(\mathcal{R}')$ is also a cycle in $PG^D(\mathcal{R})$, and since $PG^D(\mathcal{R})$ satisfies Y, $PG^U(\mathcal{R}')$ also does. Therefore, $\mathcal{R}'$ does not satisfy Y^D but satisfies Y^U. □

We also prove that strict inclusions between dependency-based conditions lead to strict inclusions between piece-unifier-based conditions.

Theorem 3.6

Let Y_1 and Y_2 be two acyclicity properties. If $Y_1^D \subset Y_2^D$ and $Y_2^D \subset Y_2^U$, then $Y_1^U \subset Y_2^U$.

Proof: Let $\mathcal{R}$ be a set of rules such that $\mathcal{R}$ satisfies Y_2^D but not Y_1^D. We rewrite $\mathcal{R}$ into $\mathcal{R}'$ as follows. For each pair of rules $R_i, R_j \in \mathcal{R}$ such that R_j depends on R_i, for each variable x in the frontier of R_j, and each variable y in the head of R_i; if x and y occur both in a given predicate position, we add to the body of R_j a new atom $p_{i,j,x,y}(x)$ and to the head of R_i a new atom $p_{i,j,x,y}(y)$ where $p_{i,j,x,y}$ denotes a fresh predicate. This construction allows each term from the head of R_i to propagate to each term from the body of R_j if they shared some predicate position in $\mathcal{R}$. Thus, any cycle in $PG^D(\mathcal{R})$ is also a cycle in $PG^U(\mathcal{R}')$, without modifying behaviour w.r.t. the acyclicity properties. Hence R' satisfies Y_2^U but does not satisfy Y_1^U. □

A further extension we can make is by doing a finer analysis of unifiers and marked cycles. We first define the notion of "incompatible" sequence of unifiers, which ensures that a given sequence of rule applications is impossible. Then, a cycle for which all sequence of unifiers are incompatible can be safely ignored. Though, we point out that the notion of piece-unifier is (again) not appropriate to such a purpose, hence we have to weaken it.

Definition 3.39 (Weak Unifier)
Given two rules $R_1 = (B_1, H_1)$ and $R_2 = (B_2, H_2)$, a weak unifier *μ of B_2 with H_1 is a piece-unifier as defined in Definition 1.29 that satisfies point (i) and (ii) but does not impose the piece condition of point (iii).*

Example 3.23 (Weak Unifier)
Consider rules $R_1 = r(x_1, y_1) \to p(z_1, y_1)$ and $R_2 = p(x_2, y_2) \wedge q(x_2) \to H_2[x_2, y_2]$. There is a weak-unifier of B_2 with H_1: $\mu_w = \{\{p(x_2, y_2)\}, \{\{z_1, x_2\}, \{y_1, y_2\}\}, \{p(z_1, y_1\}\}$, but this is not a piece-unifier. Indeed, x_2 also occurs in $q(x_2)$ and it is unified with the existential variable z_1, thus, $\{p(x_2, y_2)\}$ is not a piece.

Definition 3.40 (Compatible Unifier)
Given a set of rules $\mathcal{R}$ and two rules $R_1 = (B_1, H_1)$ and $R_2 = (B_2, H_2)$ in $\mathcal{R}$, a weak unifier $\mu = (B_2', P_\mu, H_1')$ of B_2 with H_1 is compatible *w.r.t. $\mathcal{R}$ if for each position $[a, i]$ in B_2' such that $\sigma_\mu(term([a, i]))$ is an existential variable z, $PG^U(\mathcal{R})$ contains a path from a position in which z occurs to $[a, i]$, without going through another existential position. Otherwise μ is* incompatible.

Note that in a set of rules $\mathcal{R}$, any piece-unifier of some rule body B_i with some rule head H_j is necessarily compatible with respect to $\mathcal{R}$.

Proposition 3.26

Let $\mathcal{R}$ be a set of rules, $R_1, R_2 \in \mathcal{R}$, and μ be a weak unifier of B_2 with H_1. If μ is incompatible w.r.t. $\mathcal{R}$, then no application of R_2 can use an atom in $\sigma_\mu(H_1)$.

Proof: We first formalise the sentence "no application of R_2 can use an atom in $\sigma_\mu(H_1)$" by the following sentence: "no application π' of R_2 can map an atom $a \in B_2$ to an atom b produced by an application (R_1, π) such that $b = \pi(b')$, where π and π' are more specific than μ".

Consider the application of R_1 to a set of facts F according to a homomorphism π' such that for an atom $a \in B_2$, $\pi'(a) = b = \pi(b')$, where both π and π' are more specific than μ. Note that this implies that $\mu(a) = \mu(b')$. Assume that b contains a fresh variable z_i produced from an existential variable z in H_1. Let z' be the variable

from a such that $\pi'(z') = z_i$. Since the domain of π' is the variables of B_2, all atoms from B_2 in which z' occurs at a given position $[p, j]$ are also mapped by π' to atom containing z_i in the same position $[p, j]$. Since z_i is a fresh variable, these atoms have been produced by sequences of rule applications starting from (R_1, π). Such a sequence of rule applications exists only if there is a path in PG^U from a position of z in H_1 to $[p, j]$; moreover, this path cannot go through an existential position, otherwise z_i cannot be propagated. Hence μ is necessarily compatible. □

Definition 3.41 (Unified Rule)
Given a set of rules $\mathcal{R}$ and two rules $R_1, R_2 \in \mathcal{R}$ such that there is a compatible unifier μ of B_2 with H_1, the unified rule *associated with μ, denoted by $R_\mu = R_1 \diamond_\mu R_2$, is defined by $H_\mu = \sigma_\mu(H_1) \cup \sigma_\mu(H_2)$ and $B_\mu = \sigma_\mu(B_1) \cup (\sigma_\mu(B_2) \setminus \sigma_\mu(H_1))$.*

Example 3.24 (Unified Rule)
Let $\mathcal{R}$ be the set of rules composed of the following: $R_1 = p(x_1, y_1) \rightarrow q(y_1, z_1)$, $R_2 = q(x_2, y_2) \rightarrow r(x_2, y_2)$, $R_3 = r(x_3, y_3) \wedge s(x_3, y_3) \rightarrow p(x_3, y_3)$, and $R_4 = q(x_4, y_4) \rightarrow s(x_4, y_4)$.

Let μ be the following compatible unifier of B_2 with H_1:

$$\mu = \{\{q(x_2, y_2)\}, \{q(y_1, z_1)\}, \{\{x_2, y_1\}, \{y_2, z_1\}\}\}$$

Then we obtain the unified rule associated with μ: $R_1 \diamond_\mu R_2 = p(x_1, y_1) \rightarrow q(y_1, z_1) \wedge r(y_1, z_1)$. This means that while R_3 cannot be applied right after an application of $R_1 \diamond_\mu R_2$. However the atoms needed to apply $s(x_3, y_3)$ can be brought by a sequence of rule applications (R_1, R_4). We thus relax the notion of piece-unifier to take into account arbitrary long sequences of rule applications.

Definition 3.42 (Compatible Sequence of Unifiers)
Given a set of rules $\mathcal{R}$, and a sequence of rules $(R_1, \ldots, R_{k+1})$ in $\mathcal{R}$, a sequence $s = (R_1, \mu_1, R_2, \ldots, \mu_k R_{k+1})$, where for $1 \leq i \leq k$, μ_i is a weak unifier of B_{i+1} with H_i, is a compatible sequence of unifiers *w.r.t. $\mathcal{R}$ if:*

(i) μ_1 is a compatible unifier of B_2 with H_1 w.r.t. $\mathcal{R}$, and

(ii) if $k > 0$, the sequence obtained from s by replacing (R_1, μ_1, R_2) with $R_1 \diamond_{\mu_1} R_2$ is a compatible sequence of unifiers.

Example 3.25
In Example 3.24, the sequence $(R_1, \mu_1, R_2, \mu_2, R_3, \mu_3, R_1)$ with the obvious μ_i is compatible.

We can now improve previous acyclicity properties as we did previously.

Definition 3.43 (Compatible Cycle)
Given an acyclicity property Y, and a set of rules $\mathcal{R}$, the compatible cycles *for $[a,i] \in PG^U(\mathcal{R})$ are all marked cycles C for $[a,i]$ such that there is a compatible sequence of unifiers induced by C. We say that property Y^{U+} is satisfied by $\mathcal{R}$ if for each existential position $[a,i]$, there is no compatible cycle for $[a,i]$.*

Similary results to those obtained for PG^U can be proved for PG^{U+} as state the next theorems.

Theorem 3.7

Let Y be an acylicity property. Then, $Y^U \subseteq Y^{U+}$. Moreover, if $Y^D \subset Y^U$ then $Y^U \subset Y^{U+}$.

Proof: Inclusion follows immediately from the definitions.

We now show that this inclusion is strict. Let $\mathcal{R}$ be a set of rules satisfying Y^U but not Y^D. We build a set of rules $\mathcal{R}'$ that satisfies Y^{U+} but not Y^U. To this aim, we first increase the arity of each predicate of $\mathcal{R}$ by two, and in each rule body and head, we put two fresh variables t_1 and t_2 in those positions. E.g., a rule $s(x,y) \rightarrow t(y,z)$ would become $s(x,y,t_1,t_2) \rightarrow t(y,z,t_1,t_2)$. Then, for each rule $R = (B,H)$, we create four fresh predicates p, q_1, q_2, r whose arity is respectively $|var(H)|$, 2, 2 and 2, and five fresh variables z_1, z_2, z_3, z_4 and z_5. Then we "split" R into four rules (where $\vec{x}$ is a list of all variables from H):

- $R_1 = B \rightarrow p(\vec{x}, z_1, z_2)$,
- $R_2 = p(\vec{x}, z_1, z_2) \rightarrow q_1(z_1, z_3)$,
- $R_3 = q_1(z_1, z_3) \rightarrow s(z_3, z_5)$,
- $R_4 = p(\vec{x}, z_1, z_2) \wedge q_1(z_1, z_3) \wedge q_2(z_1, z_4) \wedge s(z_3, z_5) \wedge s(z_4, z_5) \rightarrow H$.

The graph of rule dependencies of those four rules contains the following edges: (R_1, R_2), (R_2, R_3), (R_3, R_4). It can be observed that in particular, in $PG^U(\mathcal{R}')$ there is a transition edge going from the last position of the atom $p(\vec{x}, z_1, z_2)$ in rule R_1 to the last position of the "same" atom in rule R_4. The same holds for the penultimate position of these atoms. However, it can be seen that given any set of facts, rule R_4 can never be applied. But the definition of PG^U does not take this "complicated" interactions into account. Specifically, the set of rules is not Y^U anymore.

Let us now consider Y^{U+}. There is no compatible cycle in PG^U since the existential variable z_1 in rule R_1 has to go through new existential positions before reaching the position of z_1 in rule R_4. Thus, $\mathcal{R}'$ is Y^{U+}. □

Theorem 3.8

Let Y_1 and Y_2 be two acyclicity properties. If $Y_1^D \subset Y_2^D$, and $Y_2^D \subset Y_2^{U+}$, then $Y_1^{U+} \subset Y_2^{U+}$.

Proof: Observe that the transformation we used in the proof of Theorem 3.6 actually guarantees that all cycles which are present are compatible cycles. Thus, for the obtained set of rules $\mathcal{R}'$ and any acyclicity property Y, $\mathcal{R}'$ satisfies Y^U if and only if $\mathcal{R}'$ satisfies Y^{U+}. □

We now prove that if the complexity of recognition of an acyclicity property Y is in $coNP$, then, checking that a set of rules satifies one of the extended properties Y^D, Y^U or Y^{U+} is *coNP-complete*.

To this end, we make use of the following proposition.

Proposition 3.27

Given a set of rules $\mathcal{R}$ and a set of facts, if there is an $h \to b$ triggering derivation sequence (with $h \in H$ and $b \in B'$, where $R = (B, H)$ and $R' = (B', H')$ are two rules from $\mathcal{R}$), then there exist a non-empty set of paths $\mathcal{P} = \{P_1, \ldots, P_k\}$ from R in $GRD(\mathcal{R})$ such that $\sum_{1 \leq i \leq k} |P_i| \leq |\mathcal{R}| \times |terms(H)|$ and a piece-unifier of B' with the head of an agglomerated rule along $\mathcal{P}$ that unifies h and b.

Proof: The piece-unifier is entirely determined by the terms that are forced into the frontier by an fr-atom. Hence, we need to consider at most one path for each term in H. Moreover, each (directed) cycle in the GRD (that is of length at most $|\mathcal{R}|$) needs to be traversed at most $|terms(H)|$ times, since going through such a cycle without creating a new frontier variable cannot create any new unifier. Hence we need to consider only paths of polynomial length. □

We are now ready to prove the complexity of recognition.

Theorem 3.9 (Complexity of Recognition)

Let Y be an acyclicity property. If checking that a set of rules $\mathcal{R}$ satisfies Y is in $coNP$, then checking that $\mathcal{R}$ satisfies Y^D, Y^U or Y^{U+} is *coNP-complete*.

Proof: One can guess a cycle in $PG^D(\mathcal{R})$ (or $PG^U(\mathcal{R})$, or $PG^{U+}(\mathcal{R})$) such that the property Y is not satisfied by this cycle. From the previous property, each edge of the cycle has a polynomial certificate, and checking if a given substitution is a piece-unifier can also be done in polynomial time. Since Y is in $coNP$, we have a polynomial certificate that this cycle does not satisfy Y. Membership to $coNP$ follows.

The completeness part is proved by a simple reduction from the co-problem of rule dependency checking (which is thus a coNP-complete problem). Let R_1 and R_2 be two rules. We first define two fresh predicates p and s of arity $|var(B_1)|$ and two fresh predicates q and r of arity $|var(H_2)|$. We build $R_0 = p(\vec{x}) \rightarrow s(\vec{x})$ where $\vec{x}$ is a list of all variables in B_1, and $R_3 = r(\vec{x}) \rightarrow p(\vec{z}) \wedge q(\vec{x})$, where $\vec{z} = (z, z, \ldots, z)$, where z is a variable which does not appear in H_2. We rewrite R_1 into $R_1' = B_1 \wedge s(\vec{x}) \rightarrow H_1$ and R_2 into $R_2' = B_2 \rightarrow H_2 \wedge r(\vec{x})$, where $\vec{x}$ is a list of all variables in H_2. One can check that $\mathcal{R} = \{R_0, R_1', R_2', R_3\}$ contains a cycle going through an existential variable (thus, it is not wa^D) iff R_2 depends on R_1. □

Figure 3.10 summarises relationships between the different acyclicity properties. Edges are oriented from bottom to top, and an edge from an acyclicity property C_1 to an acyclicity property C_2 means that $C_1 \subset C_2$. All inclusions are strict. Furthermore, the figure is complete, that is, if there is no path between two properties, then they are incomparable.

3.5 Other Acyclicity Conditions

We now consider two other acyclicity conditions, namely MFA and MSA, that do not fit well with our framework. However, we show that MSA can be recast in our terms, hence generalised in the same way as the previously mentioned classes.

3.5.1 Model Summarizing Acyclicity and Model Faithful Acyclicity

Model Faithful Acyclicity (MFA) and Model Summarizing Acyclicity (MSA) are two properties of a set of rules ensuring the finiteness of the frontier-restricted chase [GHK+13]. Originally they were defined using the skolem chase, but as mentioned in Section 3.1, the skolem chase and the frontier-restricted chase yield isomorphic results.

The ideas underlying the verification of these properties are the following:

1. Build the critical instance $Crit(\mathcal{R})$ of the set of rules $\mathcal{R}$. It is a set of facts which ensures that any rule of $\mathcal{R}$ can be applied, and is formally defined in Definition 3.44.

2. Consider a set of rules $\mathcal{R}'$ which is either $\mathcal{R}$ (in the case of MFA) or a specialisation of $\mathcal{R}$ (in the case of MSA), where every existential variable in $\mathcal{R}$ is replaced by a fresh constant (note that those constants do not occur in $Crit(\mathcal{R})$).
3. Run the frontier-restricted chase on $Crit(\mathcal{R})$ w.r.t. $\mathcal{R}'$ until either it halts or some "existential variable cycle" (see below) is detected. This procedure is ensured to halt.

The set of rules is then MFA (resp. MSA) if this algorithm halts without detecting any existential variable cycle.

Initially, MSA has been thought as an approximation to MFA that enjoys a lower recognition (and reasoning) complexity.

Critical Instance

Definition 3.44 (Critical Instance)
Let $\mathcal{R}$ be a set of rules. The critical instance *of $\mathcal{R}$, denoted by $Crit(\mathcal{R})$ is built as follows. Consider $\mathcal{C}$ the set of all constants occurring in some rule $R \in \mathcal{R}$ (if no constant occur in the set of rules $\mathcal{C} = \{*\}$, where $*$ is a special fresh constant). For every predicate p of arity r appearing in $\mathcal{R}$, for every tuple of constants $(c_1, \ldots, c_r) \in \mathcal{C}^r$, add the atom $p(c_1, \ldots, c_r)$ to $Crit(\mathcal{R})$.*

The critical instance is sufficient to check the universal termination of the frontier-restricted chase w.r.t. a given set of rules.

Proposition 3.28 ([Mar09])

Let $\mathcal{R}$ be a set of rules. If σ_{fr}-$chase(Crit(\mathcal{R}), \mathcal{R})$ is finite then for any set of facts $\mathcal{F}$, σ_{fr}-$chase(\mathcal{F}, \mathcal{R})$ is finite.

We now define MFA and MSA in a way that differs from their original definitions but which is more adapted to our purpose.

Model Faithful Acyclicity

When building the σ_{fr}-$chase(Crit(\mathcal{R}), \mathcal{R})$, we maintain a graph of existential variables that we denote by $Evg(\mathcal{R})$. Vertices of this graph are the existential variables in $\mathcal{R}$ (without loss of generality, we assume that no two different existential variables share the same name). When we apply a rule $B \to H$ according to a homomorphism π, if H contains an existential variable z, then z is replaced in $\pi^{safe}(H)$ by a fresh variable t. We say that t is *generated* and originates from z. When we apply a rule $B \to H$ according to a homomorphism π, if π maps a term of B to a generated variable that originates from z, and there is an existential variable z' in H, then we add an edge from z to z' in $Evg(\mathcal{R})$.

Example 3.26 (MFA)
Consider the set of rules $\mathcal{R}$ composed of the single rule $R = p(x, y) \to q(y, z) \wedge p(z, t)$. Then $Crit(\mathcal{R}) = p(*, *) \wedge q(*, *)$. The two vertices of $Evg(\mathcal{R})$ are z and t. With the first application of R on $Crit(\mathcal{R})$, we add the atoms $q(*, z_0)$ and $p(z_0, t_0)$, and add no edge to $Evg(\mathcal{R})$, since the body of R is not mapped to any generated variable. With the second application of R (according to the homomorphism $\{x \mapsto z_0, y \mapsto t_0\}$), we add the atoms $q(t_0, z_1)$ and $p(z_1, t_1)$ to the set of facts, and add the edges $(z, z), (z, t), (t, z)$ and (t, t) to $Evg(\mathcal{R})$, which is now cyclic. Noe that the chase would actually be infinite.

A set of rules $\mathcal{R}$ is MFA when the frontier-restricted chase of $Crit(\mathcal{R})$ w.r.t. $\mathcal{R}$ is finite and the corresponding existential variable graph does not contain any cycle. Hence, in the previous example $\mathcal{R}$ is not MFA. When $\mathcal{R}$ is MFA, Proposition 3.28 ensures that $\mathcal{R}$ is FES. Verifying if $\mathcal{R}$ is MFA is decidable: indeed, either the frontier-restricted chase is finite, or a cycle will be generated in $Evg(\mathcal{R})$ in finite time.

Proposition 3.29 ([GHK+13])

Let $\mathcal{R}$ be a set of rules. Recognising if $\mathcal{R}$ is MFA is 2ExpTime-complete.

Model Summarizing Acyclicity

When checking if a set of rules $\mathcal{R}$ is MSA, we also consider the critical instance $Crit(\mathcal{R})$. Then we consider a set of rules $MSA(\mathcal{R})$ which is built by replacing each rule $R \in \mathcal{R}$ by a rule $MSA(R)$ in which each existential variable z is replaced by a fresh constant c_z. As for MFA, we consider the graph $Evg(\mathcal{R})$ in which vertices are existential variables in $\mathcal{R}$. If an existential variable z has been replaced by a fresh constant c_z, when c_z appears in σ_{fr}-$chase(Crit(\mathcal{R}), MSA(\mathcal{R}))$, we say that c_z is *generated* and *originates* from z. Up to these modifications, edges are added to $Evg(\mathcal{R})$ as for MFA.

Example 3.27 (MSA)
Consider again the set of rules $\mathcal{R}$ from Example 3.26, and its critical instance $Crit(\mathcal{R})$. The set of rules $MSA(\mathcal{R})$ contains the following single rule: $MSA(R) = p(x, y) \to q(y, c_z) \wedge p(c_z, c_t)$. With the first application of $MSA(R)$ on $Crit(\mathcal{R})$ we add the atoms $q(*, c_z)$ and $p(c_z, c_t)$, and we add no edge to $Evg(\mathcal{R})$. With the second application, we add the atoms $q(t_z, c_z) \wedge p(t_z, c_z)$ and the edges $(z, z), (z, t), (t, z)$ and (t, t). Though the frontier-restricted chase is finite (as opposed to Example 3.26), $\mathcal{R}$ is not MSA because of these cycles.

A set of rules $\mathcal{R}$ is MSA if when running the frontier-restricted chase of $Crit(\mathcal{R})$ w.r.t. $MSA(\mathcal{R})$ we do not generate any cycle in $Evg(\mathcal{R})$.

It has been shown in [GHK+13] that if $\mathcal{R}$ is MSA then $\mathcal{R}$ is MFA. Example 3.28 shows that this inclusion is strict.

As intended, checking MSA is simpler than checking MFA.

Proposition 3.30 ([GHK+13])

Let $\mathcal{R}$ be a set of rules. Recognising if $\mathcal{R}$ is MSA is ExpTime-complete.

Example 3.28 (MFA $\not\subseteq$ MSA)
Consider the set of rules $\mathcal{R}$ composed of the following rules:

- $R_1 = a(x) \to r(x, u) \wedge b(u)$
- $R_2 = b(x) \to s(x, v) \wedge t(v, x)$
- $R_3 = a(z) \wedge s(z, x) \to c(x)$
- $R_4 = c(z) \wedge t(z, x) \to a(x)$

The critical instance of $\mathcal{R}$ is the following:

$$Crit(\mathcal{R}) = a(*) \wedge b(*) \wedge c(*) \wedge r(*, *) \wedge s(*, *) \wedge t(*, *)$$

The vertices of $Evg(\mathcal{R})$ are u and v.

Let us check that $\mathcal{R}$ is MFA. At the first breadth-first step of the frontier-restricted chase, we apply R_1 and produce $r(*, u_1)$ and $b(u_1)$, and we apply R_2 and produce $s(*, v_1)$ and $t(v_1, *)$. No edge is generated in $Evg(\mathcal{R})$. In the second breadth-first step, we apply R_2 and produce $s(u_1, v_2)$ and $t(v_2, u_1)$, adding the edge (u, v) to $Evg(\mathcal{R})$. Then we apply R_3 and produce $c(v_1)$, adding no edge.

The chase then halts, and therefore σ_{fr}-chase$(Crit(\mathcal{R}), \mathcal{R})$ is finite. No cycle has been generated in $Evg(\mathcal{R})$, thus $\mathcal{R}$ is MFA.

Now we check that $\mathcal{R}$ is not MSA. The translation of the rules is the following:

- $MSA(R_1) = a(x) \to r(x, c_u) \wedge b(c_u)$
- $MSA(R_2) = b(x) \to s(x, c_v) \wedge t(c_v, x)$
- $MSA(R_3) = R_3$
- $MSA(R_4) = R_4$

Vertices of $Evg(\mathcal{R})$ are the same, and we consider the same critical instance as for MFA.

In the first breadth-first step, we apply R_1 and produce $r(*, c_u)$ and $b(c_u)$, and apply R_2 and produce $s(*, c_v)$ and $t(c_v, *)$. As for MFA, no edge is added to $Evg(\mathcal{R})$ yet. During the second breadth-first step, we apply R_2 and produce $s(c_u, c_v)$ and

$t(c_v, c_u)$, and we add the edge (u, v) to $Evg(\mathcal{R})$. We also apply R_3 and produce $c(c_v)$. Now, there is a third breadth-first step, in which we apply rule R_4 and produce $a(c_u)$. Finally, in the fourth breadth-first step, we apply rule R_1 again, adding the atoms $r(c_u, c_u)$ and $b(c_u)$, and creating the edge (u, u) in $Evg(\mathcal{R})$, thus $Evg(\mathcal{R})$ is cyclic. Thus, $\mathcal{R}$ is not MSA.

3.5.2 Extending Model Summarizing Acyclicity

We first point out that no marking for MFA is possible. Indeed our definition of marking function (Definition 3.28) relies on a "local" propagation, and MFA does not rely on local behaviour. However, we are able to "force" MSA into our definitions. Note that this definition is not natural, and is provided only to be able to handle MSA as the other acyclicity-based decidable rule classes.

Intuitively, the idea is to run the frontier-restricted chase on the critical instance $Crit(\mathcal{R})$ with the modified set of rules $MSA(\mathcal{R})$ as previously, but with the exception that we only add the atoms from rule applications that correspond to edges in PG^X.

Definition 3.45 (Model Summarizing Acyclicity Marking)
Given a set of rules $\mathcal{R}$, the MSA marking M of a position $[a, i]$ in $\mathcal{R}$ with respect to some transition edge condition X is the marking obtained as follows.

First, consider the critical instance $Crit(\mathcal{R})$ of $\mathcal{R}$ and translate $\mathcal{R}$ into $MSA(\mathcal{R})$. Now, run a modified frontier-restricted chase, which, when applying a rule $R = (B, H)$ according to a homomorphism π, adds each atom $h \in \pi^{safe}(H)$ only if $PG^X(\mathcal{R})$ contains an edge from an atom h' in the head of a rule R' such that there is some $b \in \pi(B)$ where b has been obtained from h' to the atom of H from which h is obtained. Then, the marking of an existential position $[a, i]$ in which variable z occurs is the set of existential positions corresponding to the existential variables z' in $Evg(\mathcal{R})$ such that there is a path from z to z'.

Proposition 3.31

A set of rules $\mathcal{R}$ is MSA iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with the MSA marking.

Proof: In the case of $PG^F(\mathcal{R})$, all positions i in atoms a and b with the same predicate are linked. Thus the condition for adding a head atom in the MSA marking is always satisfied. Therefore, the frontier-restricted chase we run in the MSA marking is exactly the same as for MSA. Since the marking of each existential position $[a, i]$ (with variable z) is exactly the set of direct or indirect successors of z in $Evg(\mathcal{R})$, the marking of $[a, i]$ contains $[a, i]$ if and only if $Evg(\mathcal{R})$ contains a cycle going through z. □

From the above proposition, Proposition 3.21 and Theorems 3.5 and 3.7, we know that $MSA \subset MSA^D \subset MSA^U \subset MSA^{U+}$.

Furthermore, it is easy to see that the complexity of recognition of these variants of MSA is still in ExpTime.

Proposition 3.32

Given a set of rules $\mathcal{R}$, checking that $\mathcal{R}$ satisfies MSA^D, MSA^U or MSA^{U+} is in ExpTime.

Proof: Checking MSA consists in running a Datalog engine while building the $Evg(\mathcal{R})$ graph. Furthermore, we can build the PG^X (for $X = D, U$ or U^+) and we have shown in Theorem 3.9 that checking if an edge if present or not in this graph is in $coNP$. Membership to $ExpTime$ follows. □

Conclusion

In this chapter, we have first formally defined the notion of chase in a generic way. Since the CQ entailment problem is undecidable, the chase of a set of facts with respect to a set of rules may be infinite.

We have then recalled various sufficient syntactic conditions that ensure finiteness of some chase variant. Thanks to a new tool, we have rewritten all these conditions in a unified way, before extending them through a finer analysis of interactions between rules.

Furthermore, since we have abstracted from the details of each condition independently, our results will also carry over new acyclicity conditions, may they be found.

Finally, the (worst-case) complexity of recognition of these new generalisations is the same as either the original condition, or the rule dependency checking, providing larger rule classes at no additional cost.

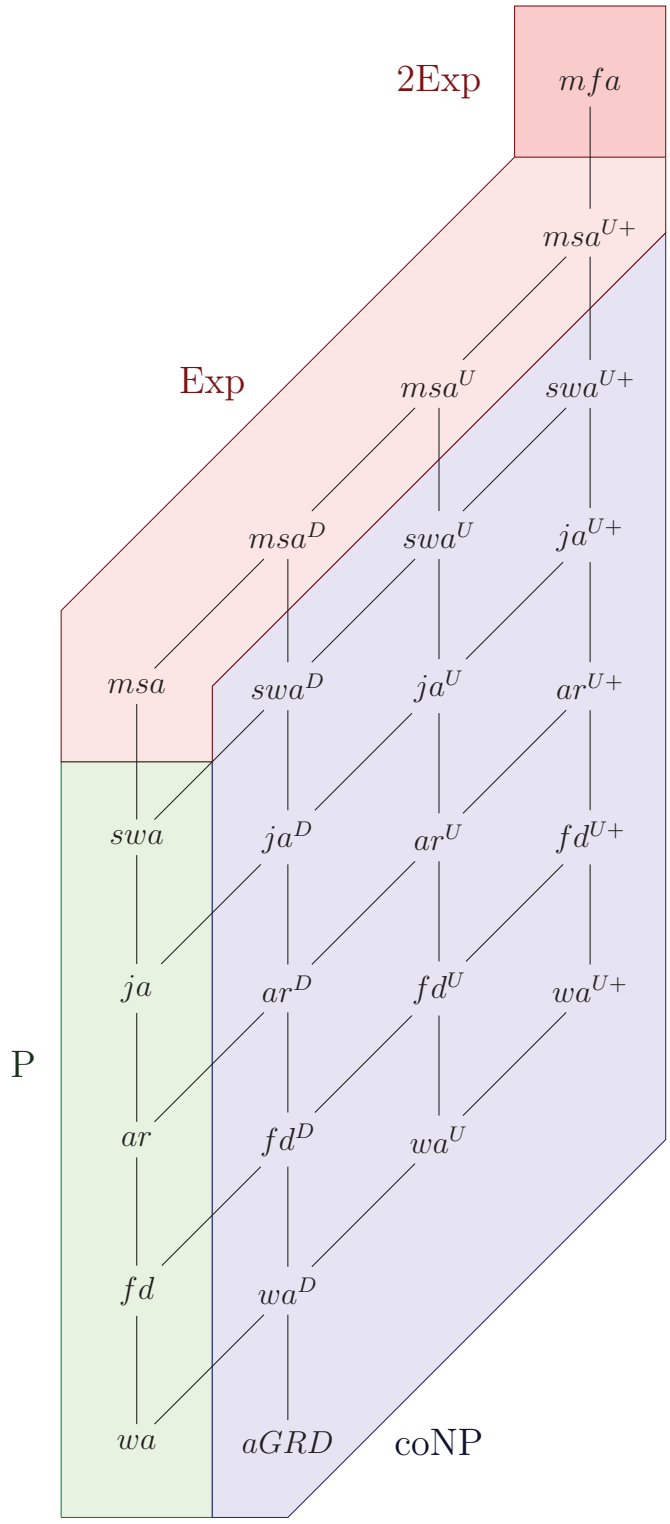

Figure 3.10: Relations between acyclicity properties with complexities of recognition

Chapter 4

Combining Transitivity and Decidable Classes of Existential Rules

Contents

Known decidable classes of existential rules are able to express many useful properties of binary relations (for instance, inverses, symmetry,...), but most of them lack the ability to define a frequently required property, namely transitivity. This limits their applicability in key application areas like biology and medecine, for which transitivity of binary relations (especially the "part of" relation) is an essential modelling construct. Hence, in this chapter we investigate the following questions: is transitivity compatible with known decidable classes of existential rules? For the cases where the answer is positive, what is the complexity of the entailment problem?

To answer these questions, we first recall various positive and negative results from the litterature, in the context of both description logics and existential rules.

Our contribution is then twofold. On the one hand we provide a general undecidability result, which in particular allows us to clarify the landscape of *FES* rule classes. On the other hand, we devise an algorithm that shows that transitivity and linear existential rules (which are GBTS and FUS) are compatible up to a minor safety condition, and analyse the complexity of the CQ entailment problem in this case.

Let us first introduce some terminology: a predicate p is said to be *transitive* in a set of rules $\mathcal{R}$ if there is some rule $R \in \mathcal{R}$, such that $R = p(x, y) \wedge p(y, z) \rightarrow p(x, z)$. Such a rule is called a *transitivity rule*. Furthermore, given a rule class C, we denote by C*+trans*, the rule class that contains all sets of rules that can be built from rules in C and transitivity rules. We extend this notation to knowledge-base and call a C*+trans* knowledge-base a knowledge-base where the set of rules belongs to the class C*+trans*. Finally, an atom using predicate p is called a p-atom.

Let us now explain intuitively why transitivity rules may be dangerous when added to a BTS or a FUS set of rules. With respect to BTS rules, adding transitivity may destroy the tree-like structure of the universal models. Indeed, transitivity rules may create links between vertices arbitrarily far away.

With respect to FUS rules, a single transitivity rule is enough to prevent the use of classical backward chaining algorithms: if an atom using a transitive predicate occurs at some point in a query rewriting, this atom can be rewritten into a path of unbounded length. Thus, *FUS* rules and transitivity may not be compatible (we show in Theorem 4.4 that, indeed, they are not).

Example 4.1 (Backward Chaining with Transitivity)
Consider a Boolean conjunctive query $Q = \exists x \exists y(s(x) \wedge p(x, y) \wedge t(y))$*, that could be read as "is there an* s *and a* t *linked by a* p*". Now if we add the transitivity rule* $R = p(x, y) \wedge p(y, z) \rightarrow p(x, z)$*, the query could be read as "is there a* **path** *from an* s *to a* t *that goes only through* p*-atoms". If we rewrite Q with R, we obtain the following rewriting:* $Q_1 = s(x) \wedge p(x, z_0) \wedge p(z_0, y) \wedge t(y)$*, which is incomparable with Q. Then we can keep going by extending the length of the path, and obviously, we will not be able to rewrite Q with R into a (finite) UCQ.*

The behaviour of transitivity regarding *FES* rules, is not that obvious, and we give later an almost complete picture of its effects on known *FES* classes.

4.1 Transitivity and BTS/FES rules

4.1.1 Overview of Known Results

While many decidable fragments of existential rules cannot express transitivity, this feature is offered by several description logics. As a matter of fact, adding transitivity to description logics often does not increase the complexity of conjunctive query entailment (exceptions are given in [ELOS09]). Though, it is known to complicate the design of query answering procedures due to the fact that it destroys the tree

structure of the logical models upon which description logics reasoning algorithms typically rely.

More specifically, in [HS99], the description logic $\mathcal{ALCH}$ has been extended to allow for transitivity axioms. More recently, in [EOS+12], the Horn-$\mathcal{SHIQ}$ description logic has been shown to be compatible with transitivity. While worst-case complexities are the same as without transitivity, the algorithms are more involved. We do not give details about the exact procedure since it would require to introduce many formal notions about description logics. It appears that these techniques heavily exploit the particular shape of DL axioms, hence do not seem to be translatable to existential rules.

At the beginning of our study (2014), little was known about combining existential rules and transitivity. Though, inclusion dependencies, which are a subclass of linear rules, and functional dependencies, which are known to "destroy" tree structures as transitivity rules do, were known to be incompatible [CV85].

In what follows, we recall various results from the literature which are all relative to the combination of GBTS rule classes with transitivity.

The first rule class of interest is that of *guarded* rules [CGK08, CGK13], which ensures that the chase of any set of facts with these rules has a tree-like shape.

Definition 4.1 (Guarded [CGK13])
A rule $R = (B, H)$ is said to be guarded *if its body contains an atom α such that for any variable x occurring in B, x appears in α (such an atom is called a* guard*). By extension, a set of rules is said to be* guarded *if all its rules are.*

It has recently been shown that combining guarded rules with transitivity leads to undecidability, even under strong syntactic restrictions [GPT13]. The theorem we state below is not exactly formulated as in this paper, but the restrictions we add can be easily derived from their proof.

Theorem 4.1 ([GPT13])

Atomic conjunctive query entailment over *guarded+trans* knowledge bases is undecidable, even when the guarded rules are restricted to the two-variable fragment of first-order logic, using only unary and binary predicates and only two transitive predicates.

In their proof, they show that it is possible to simulate a Turing Machine with only guarded and transitivity rules: the guarded rules are used to implement the transition table, and to "draw" a grid, which corresponds to the space-time diagram of the Turing Machine tape. With guarded rules alone it is not possible to "reconnect" this grid, that is, if from a given cell c we consider the cell above the cell at the right of c, it is not the same as the cell which is at the right of the cell above c. This is where transitivity rules are used: they allow to reconnect correctly the grid.

However, some positive results have also been achieved regarding a generalisation of guarded rules, namely *frontier-guarded* rules, provided that a restriction is fulfilled [ABBV16]. Informally, in a frontier-guarded rule, the guard needs not to contain all the variables from the body, but only those occurring in the frontier.

Definition 4.2 (Frontier-guarded [BLM10, BLMS11])
A rule R is said to be frontier-guarded *if its body contains an atom α such that for any variable x in the frontier of R, x occurs in α. By extension, a set of rules is said to be* frontier-guarded *if all its rules are.*

The class *frontier-guarded+trans*, and thus *guarded+trans*, with the restriction that no guard uses a transitive predicate, ensures the decidability of the CQ entailment problem, as stated in the next theorem.

Theorem 4.2 ([ABBV16])

Conjunctive query entailment over *frontier-guarded+trans* knowledge bases, where no guard uses a transitive predicate, is decidable.

Another specialisation of *frontier-guarded* rules is that of *frontier-1*, where the frontier of each rule must be of size lesser than one.

Definition 4.3 (Frontier-1 [BLMS11])
A rule R is said to be frontier-1 *if $|fr(R)| \leq 1$. By extension, a set of rules is said to be* frontier-1 *if all its rules are.*

Entailment over *frontier-1+trans* knowledge bases has been recently proven to be decidable, without any restriction, as stated by the next theorem.

Theorem 4.3 ([ABBV16])

Conjunctive query entailment over *frontier-1+trans* knowledge bases is decidable and is 2ExpTime-complete in combined complexity and PTime-complete in data complexity.

These results provide a first view of the landscape of known *BTS* rule classes, except for the case of *linear* rules. Before considering this case, we investigate the case of *FES* rules.

4.1.2 A General Undecidability Result

In the following, we show that *FES+trans* (hence *BTS+trans*) and *FUS+trans* is undecidable. More precisely, the next theorem states that *aGRD+trans* is undecidable even for atomic Boolean conjunctive queries. Since *aGRD* is both *FES* and *FUS*, this negative result holds for *FES+trans* and *FUS+trans*.

Theorem 4.4

Atomic conjunctive query entailment over *aGRD+trans* knowledge bases is undecidable, even with a single transitivity rule.

Proof: The proof is by reduction from atomic conjunctive query entailment with general existential rules (which is known to be undecidable, see Section 1.2). Let $\mathcal{R}$ be a set of rules. We first translate $\mathcal{R}$ into an *aGRD* set of rules $\mathcal{R}^a$. We consider the following new predicates: p (which will be the transitive predicate) and, for each rule $R_i \in \mathcal{R}$, predicates a_i and b_i. Each rule $R_i = (B_i, H_i)$ is translated into the following two rules:

- $R_i^1 = B_i \rightarrow a_i(\vec{x}, z_1) \wedge p(z_1, z_2) \wedge p(z_2, z_3) \wedge b_i(z_3)$
- $R_i^2 = a_i(\vec{x}, z_1) \wedge p(z_1, z_2) \wedge b_i(z_2) \rightarrow H_i$

where z_1,z_2 and z_3 are existential variables and $\vec{x}$ are the variables in B_i.

Let $\mathcal{R}^a = \{R_i^1, R_i^2 \mid R_i \in \mathcal{R}\}$, and let $GRD(\mathcal{R}^a)$ be the graph of rule dependencies of $\mathcal{R}^a$, defined as follows: the vertices of $GRD(\mathcal{R}^a)$ are in bijection with $\mathcal{R}^a$, and there is an edge from a vertex R_1 to a vertex R_2 if the rule R_2 depends on the rule R_1, i.e., if there is a piece-unifier of the body of R_2 seen as a CQ with the head of R_1 (see Proposition 3.5).

We check that for any $R_i \in \mathcal{R}$, R_i^1 has no outgoing edge and R_i^2 has no incoming edge (indeed since the z_j are existential variables, no piece-unifier exists). Hence, in $GRD(\mathcal{R}^a)$ all (directed) paths are of length less or equal to one. It follows that $GRD(\mathcal{R}^a)$ has no cycle, i.e., $\mathcal{R}^a$ is *aGRD*.

Let R^t be the rule stating that p is transitive. Let $\mathcal{R}' = \mathcal{R}^a \cup \{R^t\}$. The idea is that R^t allows to "reconnect" rules in $\mathcal{R}^a$ that correspond to the same rule in $\mathcal{R}$. For any set of facts $\mathcal{F}$ (on the original vocabulary), for any sequence of rule applications from $\mathcal{F}$ using rules in $\mathcal{R}$, one can build a sequence of rule applications from $\mathcal{F}$ using rules from $\mathcal{R}'$, and reciprocally, such that both sequences produce the same set of facts (restricted to atoms on the original vocabulary). Hence, for any $\mathcal{F}$ and Q (both on the original vocabulary), we have that $\mathcal{F}, \mathcal{R} \models Q$ iff $\mathcal{F}, \mathcal{R}' \models Q$.
□

Corollary 4.1

Atomic conjunctive query entailment over *FUS+trans* or *FES+trans* knowledge bases is undecidable.

4.1.3 Clarifying the FES Landscape

In Chapter 3, we defined various acyclicity conditions divided into two families, based on the graph of rule dependencies and on graphs of positions respectively. Figure 4.1 recalls the relations between those classes: an edge going up from a rule class C to a rule class C' means that C is more specific than C'.

Since the members of the first family generalise aGRD, they are all incompatible with transitivity. Regarding the second family, super weak-acyclicity is one of its most general member, and one can check that adding transitivity rules to a *super weakly-acyclic* set of rules $\mathcal{R}$ does not create new cycles in the SWA graph of $\mathcal{R}$, hence swa and swa*+trans* coincide as stated by the next proposition.

Proposition 4.1

The classes *super weak-acyclicity* and *super weak-acyclicity+trans* coincide. Hence, entailment over *super weak-acyclicity+trans* knowledge bases is decidable.

Proof: We do not recall the definition of super weak-acyclicity (see Definition 3.20).

Consider a super weakly-acyclic set of rules $\mathcal{R}$, and suppose we let a predicate p be transitive with a rule $R^T = p(t_0, t_1) \wedge p(t_1, t_2) \rightarrow p(t_0, t_2)$. Now assume that $\mathcal{R} \cup \{R^T\}$ is not super weakly-acyclic. It means that there is a cycle in $SWA(\mathcal{R} \cup \{R^T\})$ that is not present in $SWA(\mathcal{R})$. In particular some new edge must have been added because of the transitive rule. Let us notice that this rule is represented as a vertex in $SWA(\mathcal{R})$, but does not have any outgoing edge (indeed, only rules with existential variables may have outgoing edges). Therefore, this vertex is not used by any cycle. Hence, an edge must have been added between two rules from $\mathcal{R}$, meaning that the addition of R^T must have changed the "Move" set of some variable x. If $Move(x)$ covers t_0 (resp. t_2) then that atom position $[p(t_0, t_2), 1]$ (resp. $[p(t_0, t_2), 2]$) also belongs to $Move(x)$. Furthermore, to cover t_0 (resp. t_2), $Move(x)$ must contain some atom position $[p(x_0, x_1), 1]$ (resp. $[p(x_0, x_1), 2]$). Then, it can be observed that $p(x_0, x_1)$ and $p(t_0, t_2)$ are isomorphic; therefore, if there exists an atom q in some rule, and substitutions σ_1, σ_2 such that $\sigma_1(q) = \sigma_2(p(t_0, t_2))$, then the substitution $\sigma_3 = \{x_0 \mapsto t_0, x_1 \mapsto t_2\} \circ \sigma_2$ is such that $\sigma_1(q) = \sigma_3(p(x_0, x_1))$. Therefore for any variable t from q such that $In(t)$ is covered by $Move(x)$ in $\mathcal{R} \cup \{R^T\}$, $In(t)$ is also

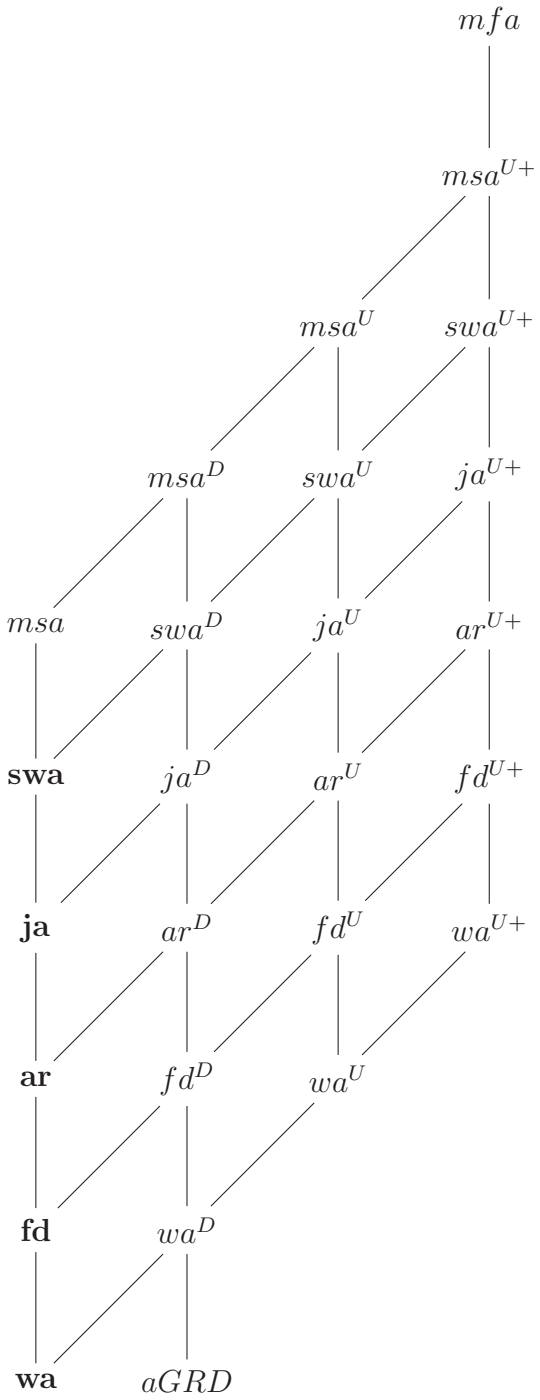

Figure 4.1: The landscape of C+*trans* classes for $C \in FES$. Decidable C+*trans* are in bold. All the other C+*trans* classes are undecidable even for atomic CQs.

covered by $Move(x)$ in $\mathcal{R}$. Hence the set of edges between rules from $\mathcal{R}$ in $\mathcal{R} \cup \{R^T\}$

is the same as in $\mathcal{R}$.

□

To complete the picture (see Figure 4.1), it remains to study the case of MSA, that generalises swa but not aGRD. We show below that MSA+*trans* is indeed undecidable even for atomic queries.

Theorem 4.5

Atomic conjunctive query entailment over MSA+*trans* knowledge bases is undecidable, even with a single transitivity rule.

Proof: The proof is by reduction from atomic conjunctive query entailment with general existential rules (which is known to be undecidable, see Section 1.2). Let $\mathcal{F}$ be a set of facts, $\mathcal{R}$ be a set of rules and Q be an atomic conjunctive query.

First we consider a new transitive predicate p, which is the only transitive predicate we use.

We rewrite $\mathcal{F}$ into $\mathcal{F}'$ as follows. For each term $t \in terms(\mathcal{F})$, we add the atoms $p(t, a_t)$ and $p(a_t, t)$ to $\mathcal{F}'$, where a_t is a fresh constant.

Then, we rewrite $\mathcal{R}$ into a MSA set of rules $\mathcal{R}^m$. For each rule $R = B \to H$, we consider the rule $R' = B' \to H'$ obtained as follows. Its body B' is composed of the atoms of B as well as the atoms $p(t, t)$ for each term $t \in terms(B)$. Its head H' contains the atoms of H as well as two atoms $p(z, x_z)$ and $p(x_z, z)$, where x_z is a fresh variable, for each existential variable z in H.

We check that $\mathcal{R}^m$ is MSA. Consider the set of rules $MSA(\mathcal{R}^m)$ as defined in Section 3.5.1 (i.e., each existential variable z in a rule head is replaced by a fresh constant c_z). If z is an existential variable of a rule $R \in \mathcal{R}$, the head of the corresponding rule R' in $\mathcal{R}^m$ contains the atoms $p(z, x_z)$ and $p(x_z, z)$, and thus the head of $MSA(R')$ contains the atoms $p(c_z, c_{x_z})$ and $p(c_{x_z}, c_z)$. We point out that, during the chase, no term t in any rule body can be mapped to c_z (because $p(c_z, c_z)$ will never appear). So the existential variable graph $Evg(\mathcal{R})$ does not contain any edge. Thus $\mathcal{R}^m$ is MSA.

Now, let R^T be the rule expressing the transitivity of p. It is clear that $(\mathcal{F}, \mathcal{R}) \models Q$ if and only if $(\mathcal{F}', \mathcal{R}^m \cup \{R^T\}) \models Q$.

We conclude that atomic conjunctive query entailment over MSA+*trans* knowledge bases is undecidable. □

These three results give us a clear picture of the combination of *FES* sets of rules with transitivity for all classes pictured in Figure 4.1. Indeed, either the class C generalises aGRD or MSA and entailment over C+*trans* knowledge-bases is undecidable even in the case of atomic queries, or C+*trans* = C.

In contrast, the behaviour of concrete *FUS* classes when combined with transitivity is still unknown (except for aGRD), and in the next section, we study the

particular case of *linear* rules, a subclass of *guarded* rules, that also enjoys the *FUS* property.

4.2 Linear Rules and Transitivity

In this section we focus on combining transitivity with linear existential rules. We first recall the definition of linear rules. Without loss of generality (if the predicate arity is unbounded) we consider rules with atomic head and no constant, and to avoid any misunderstanding, the following definition takes this into account.

Definition 4.4 (Linear Rule)
Given a rule $R = (B, H)$, we say that R is a linear *rule if $|B| = |H| = 1$, and R contains no constant. By extension, a set of rules is said to be* linear *if all its rules are.*

Linear rules are a subclass of guarded rules, and as such, are BTS. Moreover they are FUS, and thus are typically handled through backward chaining. Unfortunately, transitivity rules immediately break this property, therefore classical query rewriting is not possible. However, in this section, we propose a rewriting algorithm, that does not produce a union of conjunctive queries, but a Datalog program (i.e., a finite set of Datalog rules added with a union of conjunctive queries). This algorithm shows that, up to a minor safety condition, entailment over *linear+trans* knowledge bases is decidable. We also point out that this restriction can be lifted if we only consider atomic conjunctive queries or binary predicates.

4.2.1 Framework

In order to obtain finite representations of sets of rewritings, we first define a framework based on the notion of *pattern*.

To each transitive predicate, we assign a *pattern name*. Each pattern name has an associated *pattern definition* $P := a_1 | \dots | a_k$ where each a_i is an atom that contains the special variables #1 and #2. These variables are placeholders for the real terms that will occur in the set of atoms that use this pattern. A *pattern* is either a *standard pattern*, denoted by $P[t_1, t_2]$, or a *repeatable pattern*, denoted by $P^+[t_1, t_2]$, where P is a pattern name, and t_1 and t_2 are terms.

Definition 4.5 (UPCQ and PCQ)
A union of patterned conjunctive queries *(UPCQ) is a pair $(\mathbb{Q}, \mathbb{P})$, where $\mathbb{Q}$ is a disjunction of conjunctions of atoms and patterns, and $\mathbb{P}$ is a set of pattern definitions that gives a unique definition to each pattern name occurring in $\mathbb{Q}$.*

If there is a single conjunction of atoms $\mathcal{Q} \in \mathbb{Q}$, $(\mathbb{Q}, \mathbb{P})$ is simply called a patterned conjunctive query *(PCQ).*

Example 4.2 (UPCQ and PCQ)
The pair $(\mathbb{Q}, \mathbb{P})$ *with* $\mathbb{Q} = \mathcal{Q}_1 \vee \mathcal{Q}_2$, $\mathcal{Q}_1 = P_1^+[a, b] \wedge s_1(a, b)$, $\mathcal{Q}_2 = P_1^+[a, z] \wedge P_2^+[z, b] \wedge s_1(a, b)$, *and where* $\mathbb{P}$ *contains the following pattern definitions:*

- $P_1 := p_1(\#1, \#2)|s_2(\#1, y, \#2)$
- $P_2 := p_2(\#1, \#2)|s_2(\#2, y, \#1)$;

is a UPCQ.

The pairs $(\mathcal{Q}_1, \mathbb{P})$ *and* $(\mathcal{Q}_2, \mathbb{P})$ *are PCQs.*

In the following we use three different styles to distinguish between a UPCQ ($\mathbb{Q}$), a PCQ ($\mathcal{Q}$) and a (U)CQ (Q). Furthermore, for the sake of simplicity, we will often denote a (U)PCQ by its first component $\mathbb{Q}$, leaving the pattern definitions implicit.

With UPCQs, we are able to represent a set of rewritings in a compact way, but to this end, we first need to define the notion of an instantiation of a UPCQ.

Definition 4.6 (Instantiation of a UPCQ)
Given a UPCQ $(\mathbb{Q}, \mathbb{P})$, *an instantiation* T *of* $(\mathbb{Q}, \mathbb{P})$ *is a rooted vertex-labelled tree that satisfies the following four conditions:*

(i) the root of T *if labelled by* $\mathcal{Q} \in \mathbb{Q}$;

(ii) the children of the root are labelled by the patterns and atoms occurring in $\mathcal{Q}$;

(iii) each vertex that is labelled by a repeatable pattern $P^+[t_1, t_2]$ *may be expanded into* $k \geq 1$ *children labelled respectively by* $P[t_1, x_1], P[x_1, x_2], \ldots, P[x_{k-1}, t_2]$ *where all* x_i *are fresh variables;*

(iv) each vertex labelled by a standard pattern $P[t_1, t_2]$ *may be expanded into a single child whose label is obtained from an atom in the pattern definition of* P *in* $\mathbb{P}$ *by substituting* $\#1$ *(resp.* $\#2$*) by* t_1 *(resp.* t_2*), and freshly renaming the other variables.*

The *instance* associated with an instantiation is the PCQ obtained by taking the conjunction of its leaves. An instance of a UPCQ is an instance associated with one of its instantiations. We say that an instance is *full* when it does not contain any pattern, and we denote by $full(\mathbb{Q}, \mathbb{P})$ the set of all full instances of $(\mathbb{Q}, \mathbb{P})$.

Example 4.3 (Instantiations and instances)
Consider the UPCQ $(\mathbb{Q}, \mathbb{P})$ *from Example 4.2. Figure 4.2 depicts a first possible instantiation* T_1 *of* $(\mathbb{Q}, \mathbb{P})$ *that gives rise to the (non-full) instance* $Q_1 = P_1[a, z] \wedge P_2[z, x_1] \wedge P_2[x_1, b] \wedge s_1(a, b)$. *By expanding the three vertices labelled by patterns according to their definitions in* $\mathbb{P}$, *we may obtain the instantiation* T_2 *depicted by Figure 4.3, whose associated instance* $Q_2 = s_2(a, y_0, z) \wedge s_2(x_1, y_1, z) \wedge p_2(x_1, b) \wedge s_1(a, b)$ *is a full instance of* $(\mathbb{Q}, \mathbb{P})$.

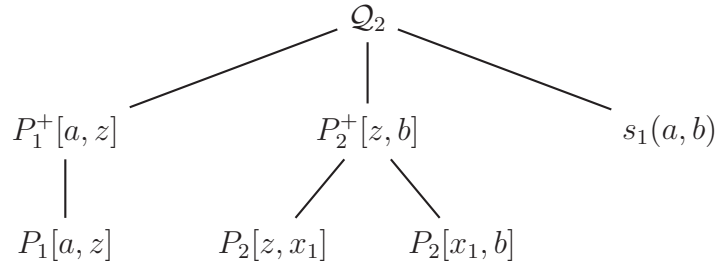

Figure 4.2: Instantiation T_1 from Example 4.3

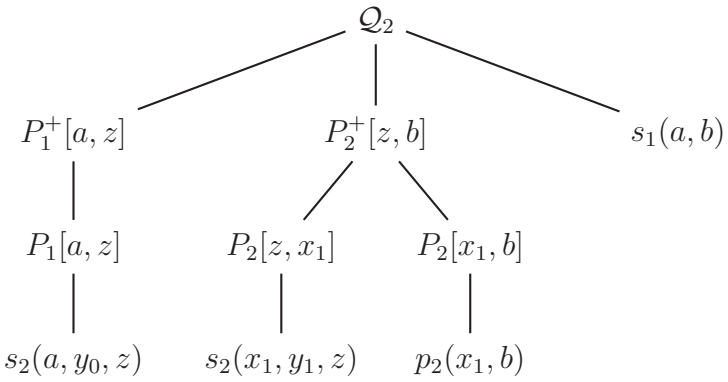

Figure 4.3: Instantiation T_2 from Example 4.3

A UPCQ $(\mathbb{Q}, \mathbb{P})$ can be translated into a set of Datalog rules $\Pi_{\mathbb{P}}$ and a UCQ $Q_{\mathbb{Q}}$ as follows. For each definition $P := a_1(\vec{t_1})| \dots |a_k(\vec{t_k})$, we create the transitivity rule $p^+(x,y) \wedge p^+(y,z) \to p^+(x,z)$ and the rules $a_i(\vec{t_i}) \to p^+(\#1,\#2)$. The UCQ $Q_{\mathbb{Q}}$ is obtained from $\mathbb{Q}$ by replacing each repeatable pattern $P^+[t_1,t_2]$ by the atom $p^+(t_1,t_2)$. Observe that $\Pi_{\mathbb{P}}$ is non-recursive except for the transitivity rules. Moreover, for any set of facts $\mathcal{F}$, the saturation of $\mathcal{F}$ with $\Pi_{\mathbb{P}}$ can be decomposed into two steps:

1. Saturation with the rules of form $a_i(\vec{t_i}) \to p^+(\#1,\#2)$;
2. Saturation with the transitivity rules (i.e., computation of the transitive closure for each transitive predicate).

Note that these two steps only produce atoms with predicates p^+.

Example 4.4 (Translation to Datalog and UCQ)
Consider the UPCQ from Example 4.2. The pattern definitions are translated into the following rules:

- $p_1(\#1,\#2) \to p_1^+(\#1,\#2)$

- $s_2(\#1, y, \#2) \rightarrow p_1^+(\#1, \#2)$
- $p_2(\#1, \#2) \rightarrow p_2^+(\#1, \#2)$
- $s_2(\#2, y, \#1) \rightarrow p_2^+(\#1, \#2)$

These four rules together with transitivity rules on p_1^+ and p_2^+ form $\Pi_{\mathbb{P}}$. Moreover, $Q_{\mathbb{Q}} = (p_1^+(a, b) \wedge s_1(a, b)) \vee (p_1^+(a, z) \wedge p_2^+(z, b) \wedge s_1(a, b))$.

We are now able to summarise informally the main ideas underlying our framework:

1. A PCQ $(\mathcal{Q}, \mathbb{P})$ represents an infinite number of CQs, i.e., all the full instances that can be obtained from $\mathcal{Q}$ using the pattern definitions in $\mathbb{P}$. A UPCQ represents all instances associated with its PCQs.
2. A UPCQ is considered entailed by a set of facts if one of its full instances is.
3. Given a UPCQ $(\mathbb{Q}, \mathbb{P})$, the pattern definitions in $\mathbb{P}$ can be translated into a set of Datalog rules $\Pi_{\mathbb{P}}$ and the query $\mathbb{Q}$ itself into a UCQ $Q_{\mathbb{Q}}$ such that $\mathbb{Q}$ is entailed by the set of facts $\mathcal{F}$ if the UCQ $Q_{\mathbb{Q}}$ is entailed by $\mathcal{F}$ saturated with rules from $\Pi_{\mathbb{P}}$, or equivalently if $Q_{\mathbb{Q}}$ is entailed by the knowledge-base $(\mathcal{F}, \Pi_{\mathbb{P}})$.

These ideas are formally stated by the next proposition.

Proposition 4.2

Let $\mathcal{F}$ be a set of facts and $(\mathbb{Q}, \mathbb{P})$ be a UPCQ; then, $(\mathcal{F}, \Pi_{\mathbb{P}}) \models Q_{\mathbb{Q}}$ iff there is some $Q \in full(\mathbb{Q}, \mathbb{P})$ such that $\mathcal{F} \models Q$.

Proof: We successively prove the two directions.

($\Leftarrow$) Let T be an instantiation of $(\mathbb{Q}, \mathbb{P})$ such that there exists a homomorphism π from its associated full instance to the set of facts $\mathcal{F}$. Let us consider a vertex of T labelled by a standard pattern $P[t_1, t_2]$. The label $r(\rho(\vec{t}))$ of its child was obtained by choosing an atom $r(\vec{t})$ in the pattern definition of P (with ρ being the substitution used to create this child). Thus the Datalog program $\Pi_{\mathbb{P}}$ contains the rule $r(\vec{t}) \rightarrow p^+(\#1, \#2)$ where $\#1, \#2 \in \vec{t}$. By applying the rule according to the homomorphism $\pi \circ \rho$, we can add the atom $p^+(t_1, t_2)$ to $\mathcal{F}$, and let $\mathcal{F}'$ be the obtained set of facts. Let us repeat this procedure for every vertex of T labelled by a standard pattern. Consider next a repeatable pattern labelled by $P^+[t, t']$ whose children are respectively labelled by $P[t = t_1, t_2], P[t_2, t_3], \ldots, P[t_{k-1}, t_k = t']$. According to the previous rule applications, $\mathcal{F}'$ contains the atoms $p^+(t = t_1, t_2), p^+(t_2, t_3), \ldots, p^+(t_{k-1}, t_k = t')$. Then, by successive applications of the rule in $\Pi_{\mathbb{P}}$ expressing the transitivity of p^+, we finally add to $\mathcal{F}'$ the atom $p^+(t, t')$. Let $\mathcal{F}''$

be the obtained set of facts. Repeat this procedure for every vertex of T labelled by a repeatable pattern. Now the root of T is labelled by some $\mathcal{Q} \in \mathbb{Q}$. The UCQ $Q_{\mathbb{Q}}$ contains a CQ Q that was obtained from $\mathcal{Q}$ by replacing each repeatable pattern $P^+[t_1, t_2]$ by $p^+(t_1, t_2)$. Observe that the restriction of π to the terms of Q is a homomorphism from Q to $\mathcal{F}''$.

($\Rightarrow$) Conversely, let us consider a set of facts $\mathcal{F}''$ obtained by saturating the initial set of facts $\mathcal{F}$ with the rules of $\Pi_{\mathbb{P}}$, and a homomorphism π from some CQ Q' in the UCQ $\mathbb{Q}$ to $\mathcal{F}''$. Let us now build an instantiation T of $(\mathbb{Q}, \mathbb{P})$ whose full instance can be mapped to $\mathcal{F}$ thanks to a homomorphism π'. The root vertex of T is labelled by the PCQ $\mathcal{Q}$ in $\mathbb{Q}$ from which Q' was obtained. Its children are labelled by the atoms and patterns of $\mathcal{Q}$. For all terms x from $\mathcal{Q}$, we define $\pi'(x) = \pi(x)$. Let us consider a child of the root labelled by a repeatable pattern $P^+[t, t']$. It follows that $p^+(\pi(t), \pi(t'))$ is an atom of $\mathcal{F}''$. Since this atom is not in the initial set of facts, it has been obtained either by a rule associated with a pattern definition or by a sequence of applications of the rule expressing the transitivity of p^+. In this latter case, consider a path of p^+-atoms $\pi(t) = t_1, \ldots, t_k = \pi(t')$ such that no atom $p^+(t_i, t_{i+1})$ in $\mathcal{F}'$ has been obtained by a transitivity rule. Then the vertex labelled by $P^+[t, t']$ has $k+1$ children respectively labelled by $P[t = x_1, x_2], \ldots, P[x_{k-1}, x_k = t']$. For the fresh variables $x_2, \ldots, x_{k-1}$, we define $\pi'(x_i) = t_i$. Repeat this procedure for every repeatable pattern in T. Let us next consider a vertex of T labelled by a standard pattern $P[x, x']$. Since that vertex was obtained in the previous phase, we know that the atom $p^+(\pi'(x), \pi'(x'))$ is in $\mathcal{F}''$, and that it was not obtained from the application of a transitivity rule. Thus, the Datalog rule used to produce that atom is necessarily a rule obtained from the definition of the pattern P. Let $r(\vec{t}) \rightarrow p^+(\#1, \#2)$, where $\#1, \#2 \in \vec{t}$, be that rule. According to that pattern definition, we can add to the vertex labelled by $P[x, x']$ a child labelled by $r(\rho(\vec{t}))$. Since the Datalog rule was applied according to a homomorphism π'', we define, for every fresh variable $\rho(t)$, $\pi'(\rho(t)) = \pi''(t)$. Do the same for every standard pattern of T. The instance associated with T is full, and π' is a homomorphism from this full instance to the initial set of facts $\mathcal{F}$. □

We then extend the definition of piece-unifier (see Definition 1.29) to PCQs (and simply call it "unifier").

Definition 4.7 (Unifier)
A unifier $\mu = (Q', H, P_\mu)$ *of a PCQ with a rule* $R = (B, H)$ *is a single-piece unifier of one of its (possibly non-full) instances such that* Q' *is a set of (usual) atoms.*

Note that since in this section we only consider atomic-headed rules, H is composed of a single atom. Furthermore, since we consider only single-piece unifiers, Q' is ensured to be connected, which will be important in the following proofs.

We distinguish between two types of unifiers (internal and external), defined next.

Let T be an instantiation whose associated instance is Q, and $\mu = (Q', H, P_\mu)$ be a unifier of Q with some rule $R = (B, H)$. Assume T contains a repeatable pattern $P^+[t_1, t_2]$ expanded into $P[u_0, u_1], \ldots, P[u_k, u_{k+1}]$, where $u_0 = t_1$ and $u_{k+1} = t_2$. We call $P[u_i, u_{i+1}]$ *relevant for* μ if it is expanded into an atom from Q'. Because we consider only single-piece unifiers, it follows that if such relevant patterns exist, they form a "connected" sequence $P[u_i, u_{i+1}], \ldots, P[u_{j-1}, u_j]$. Terms u_i and u_j are called *external* to $P^+[t_1, t_2]$ with respect to μ; the other terms occurring in the sequence are called *internal*.

Definition 4.8 (Internal/External Unifier)
Given a unifier $\mu = (Q', H, P_\mu)$ of a PCQ $\mathcal{Q}$ with some rule $R = (B, H)$, we say that μ is internal *if all atoms from Q' are expanded from a single repeatable pattern, and no external terms are unified together or with an existential variable; otherwise μ is called* external.

Example 4.5 (Internal/External Unifiers)
Consider T_2 (depicted by Figure 4.3) whose associated instance is $Q_2 = s_2(a, y_0, z) \wedge s_2(x_1, y_1, z) \wedge p_2(x_1, b) \wedge s_1(a, b)$ from Example 4.3 and the rules:

- $R_1 = s_1(x', y') \rightarrow p_2(x', y')$,
- $R_2 = s_1(x', y') \rightarrow s_2(x', y', z')$.

The unifier of Q_2 with R_1 that unifies $p_2(x_1, b)$ with $p_2(x', y')$ is internal; whereas the unifier of Q_2 with R_2 that unifies $\{s_2(a, y_0, z), s_2(x_1, y_1, z)\}$ with $s_2(x', y', z')$ is external because it involves two repeatable patterns.

We point out that when Q is composed only of usual atoms, i.e., its single (full) instance is the query itself, the notions of external unifiers and single piece-unifiers coincide since no pattern is involved.

It is crucial for the following to understand that unifiers of PCQs are not performed on the PCQs themselves but on instances of PCQs (actually, we will consider so-called "minimally-unifiable instances" which may not be "instances" strictly speaking but are still obtained by developing PCQs).

Hence, we distinguish between a "classical direct rewriting" of an instance of a PCQ (with respect to a single-piece unifier, which is a unifier of the PCQ), and a "direct rewriting" of a PCQ (with respect to an internal or external unifier), which involves selecting a particular instance and computing a classical direct rewriting of this instance.

Moreover, as detailed in the two next sections, internal and external unifiers will be used in two different ways. Internal unifiers will be used to rewrite pattern definitions: in this case, the considered PCQ will be the atomic query $P^+[x, y]$ for a pattern name P. Hence, this kind of rewriting will be performed independently from the actual input query, which will be rewritten using only external unifiers.

Finally, in order to later "ignore" some queries during the rewriting steps, we need a way to compare unifiers, and the next definition provide us such a tool.

Definition 4.9 (Comparison between Unifiers)
Given two unifiers $\mu_1 = (Q_1, H, P_{\mu_1})$ *and* $\mu_2 = (Q_2, H, P_{\mu_2})$*, we say that* μ_2 *is* more general than μ_1*, denoted by* $\mu_2 \geq \mu_1$*, if there is a substitution* π *from* $\sigma_2(Q_2)$ *to* $\sigma_1(Q_1)$ *such that* $h(\sigma_2(Q_2)) \subseteq \sigma_1(Q_1)$ *(i.e.,* π *is a homomorphism from* $\sigma_2(Q_2)$ *to* $\sigma_1(Q_1)$*), and for all terms* x *and* y *in* $Q_2 \cup H$*, if* $\sigma_2(x) = \sigma_2(y)$ *then* $\sigma_1(\pi(x)) = \sigma_1(\pi(y))$*, where* σ_1 *and* σ_2 *are substitutions respectively associated with* P_{μ_1} *and* P_{μ_2}*.*

4.2.2 Overview of the Algorithm

The framework being defined, we can now present the global algorithm, which takes as input a Boolean conjunctive query Q and a set of rules $\mathcal{R} = \mathcal{R}_L \cup \mathcal{R}_T$, where $\mathcal{R}_L$ is a set of (atomic-headed) linear rules (with no constant), and $\mathcal{R}_T$ a set of transitivity rules; and builds a finite set of Datalog rules and a (possibly infinite) set of conjunctive queries.

The main steps of the algorithm are outlined below. The notions of internal and external rewritings will be defined in the next section.

Step 1 For each predicate p that appears in $\mathcal{R}_T$, create a pattern definition $P := p(\#1, \#2)$, where P is a fresh pattern name. The resulting set of definitions is called $\mathbb{P}_0$.

Step 2 Let $\mathcal{R}_L^+$ be the set of rules obtained from $\mathcal{R}_L$ by replacing each body atom $p(t_1, t_2)$ in a rule such that p is a transitive predicate by the repeatable pattern $P^+[t_1, t_2]$.

Step 3 (Internal Rewriting) Initialise $\mathbb{P}$ to $\mathbb{P}_0$, and repeat the following operation until fixpoint: select a pattern definition $P \in \mathbb{P}$ and a rule $R \in \mathcal{R}_L^+$ and compute the direct rewriting of $\mathbb{P}$ with respect to P, R and an internal unifier.

Step 4 Replace in Q all atoms $p(t_1, t_2)$ such that p is a transitive predicate by the repeatable pattern $P^+[t_1, t_2]$ and denote the result by $\mathcal{Q}^+$.

Step 5 (External Rewriting) Initialise $\mathbb{Q}$ to $\{\mathcal{Q}^+\}$ and repeat the following operation until fixpoint: select $\mathcal{Q}_i \in \mathbb{Q}$, compute a direct rewriting of $\mathcal{Q}_i$ with respect to $\mathbb{P}$, a rule from $\mathcal{R}_L^+$ and an external unifier; and add the result to $\mathbb{Q}$ if it is not isomorphic to some $\mathcal{Q}_j \in \mathbb{Q}$.

Step 6 Let $\Pi_{\mathbb{P}}$ be the Datalog translation of $\mathbb{P}$ and $Q_{\mathbb{Q}}$ be the possibly infinite set of conjunctive queries obtained by replacing each repeatable pattern $P^+[t_1, t_2]$ in $\mathbb{Q}$ by $p^+(t_1, t_2)$.

In a nutshell, Step 3 is always guaranteed to terminate, while Step 5 is not, however in Section 4.2.4, we propose a modification to Step 5 that ensures termination

and formulate sufficient conditions that preserve completeness. When $Q_{\mathbb{Q}}$ is finite (i.e., it is a UCQ), it can be evaluated over the set of facts saturated by $\Pi_{\mathbb{P}}$, or alternatively translated into a set of Datalog rules and passed to a Datalog engine for evaluation. Observe that the construction of $\Pi_{\mathbb{P}}$ is query-independent and can be executed as a preprocessing step.

4.2.3 Rewriting Steps

A PCQ that contains a repeatable pattern has an infinite number of instances. Instead of considering all instances of a PCQ, we consider a finite set of "instances of interest" with respect to a given rule. Such instances will be used for both the internal and external rewriting steps.

Definition 4.10 (Instantiation and Instance of Interest)
Given a PCQ $(\mathcal{Q}, \mathbb{P})$ *and rule* $R \in \mathcal{R}_L^+$ *with head predicate* p*, the* instantiations of interest *of* $(\mathcal{Q}, \mathbb{P})$ *w.r.t.* R *are constructed as follows. For each repeatable pattern* $P_i^+[t_1, t_2]$ *in* $\mathcal{Q}$*, let* $a_1^i, \ldots, a_{n_i}^i$ *be the atoms in the definition of* P_i *with predicate* p*. If* $n_i > 0$*, then expand* $P_i^+[t_1, t_2]$ *into* k *standard patterns, where* $0 \leq k \leq min(arity(p), n_i) + 2$*, and expand each of these standard patterns in turn into some* a_ℓ^i*. An* instance of interest *is the instance associated with an instantiation of interest.*

Hence, in an instance of interest with respect to a rule with head predicate p, repeatable patterns with p-atoms occurring in their definitions may have been expanded into standard p-atoms. Since the aim is to unify this instance of interest with a p-atom from a rule head, only p-atoms from the pattern definitions are considered. The upper bound on the size of these instances of interest is explained in the proof of Proposition 4.3.

Example 4.6 (Instance of Interest)
Consider again the instantiation T_2 *of* $\mathcal{Q}_2$ *pictured in Figure 4.3, and the rule* $R_2 = s_1(x', y') \rightarrow s_2(x', y', z')$ *from Example 4.5. The instance* Q_2 *associated with* T_2 *is not an instance of interest of* $\mathcal{Q}_2$ *w.r.t.* R_2*, since* $P_2[x_1, b]$ *is expanded into* $p_2(\#1, \#2)$ *whereas the head predicate of* R_2 *is* s_2 *(see the pattern definition* P_2 *in Example 4.2). If we expand* $P_2[x_1, b]$ *with* $s_2(\#2, y, \#1)$ *instead, we obtain the instance of interest* $Q_3 = s_2(a, y_0, z) \wedge s_2(x_1, y_1, z) \wedge s_2(b, y_2, x_1) \wedge s_1(a, b)$*.*

Note that while in this example, the instances of interest we build are full, it is not always the case, in particular it is possible to have an instantiation of interest that does not expand any repeatable pattern.

We next show that the set of unifiers computed on the instances of interest of a PCQ "captures" the set of unifiers associated with all its instances.

Proposition 4.3

Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ and $R \in \mathcal{R}_L^+$. For every instance Q of $(\mathcal{Q}, \mathbb{P})$ and unifier μ of Q with R, there exist an instance of interest Q' of $(\mathcal{Q}, \mathbb{P})$ w.r.t. R and a unifier μ' of Q' with R such that μ' is more general than μ.

Proof: Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ, $R \in \mathcal{R}_L^+$ be a rule with head $p(\vec{t})$, Q be an instance of $(\mathcal{Q}, \mathbb{P})$, and $\mu = (Q_2, p(\vec{t}), P_u)$ be a unifier of Q with R.

Consider a repeatable pattern $P^+[t_1, t_2]$ in the instantiation associated with Q from which some atoms in Q_2 are expanded. If $P^+[t_1, t_2]$ is expanded into $k \leq arity(p) + 2$ standard patterns in Q, [1] then there is an instance of interest Q' that expands $P^+[t_1, t_2]$ into exactly k standard patterns such that there is an isomorphism π between the atoms expanded under $P^+[t_1, t_2]$ in Q and in Q'. Assume instead that $P^+[t_1, t_2]$ is expanded in Q into $k > arity(p) + 2$ standard patterns. We denote by σ a substitution associated with P_u, and by $P[t_1 = x_0, x_1], P[x_1, x_2], \ldots, P[x_{k-2}, x_{k-1}], P[x_{k-1}, x_k = t_2]$ the sequence of standard patterns expanded from $P^+[t_1, t_2]$ in Q, and let x_s and x_e $(s < e)$ be the external terms of $P^+[t_1, t_2]$ w.r.t. μ. The unifier is single-piece, thus, for every $s < i < e$, $\sigma(x_i) = \sigma(z_i)$ for some existential variable z_i from the head of R.

We construct an instance of interest Q' and function π as follows. Starting from $\mathcal{Q}$, we expand every repeatable pattern $P^+[t_1, t_2]$ that is relevant for μ into $e - s$ standard patterns (where e and s are defined as above and depend on the particular pattern):

$$P[t_1 = x'_s, x'_{s+1}], P[x'_{s+1}, x'_{s+2}], \ldots, P[x'_{e-1}, x'_e = t_2].$$

Then for every $s \leq i \leq e$, we set $\pi(x'_i) = x_i$, and we expand $P[x'_i, x'_{i+1}]$ into the atom a'_i that is obtained from the atom a_i expanded under $P[x_i, x_{i+1}]$ in Q by replacing every x_j with x'_j. If $e - s \leq arity(p) + 2$, we are done. Otherwise, we will need to remove some standard patterns in order to satisfy the definition of instances of interest. To this end, we define a sequence $s < i_1 < j_1 < \cdots < i_m < j_m < e$ of indices as follows:

- We call $i < j$, with $s < i < j < e$, a *matching pair* in a_i and in a_j, x'_{i+1} and x'_j occur at the same position of p (hence, $\sigma(x_{i+1}) = \sigma(x_j)$);
- We say that a matching pair $i < j$ is *maximal w.r.t. index* ℓ if the following conditions hold:
  - $i \geq \ell$,
  - there is no matching pair $i' < j'$ with $\ell \leq i' < i$

[1] Strictly speaking, we mean the instantiation underlying Q, but to simplify the notation, here and later in the various proofs, we will often refer to instances, leaving the instantiation implicit.

  - there is no matching pair $i < j'$ with $j' > j$

- We let $i_1 < j_1$ be the matching pair that is maximal w.r.t. index $s + 1$
- For any k, if $i_k < j_k$ is already defined, then we let $i_{k+1} < j_{k+1}$ be the matching pair that is maximal w.r.t. index j_k, if such a pair exists (otherwise, $i_k < j_k$ is the final pair in the sequence).

Now remove from Q' all the patterns $P[x'_\ell, x'_{\ell+1}]$ such that $i_g < \ell < j_g$ for some $1 \leq g \leq m$, as well as the atoms that are expanded from such patterns. We claim that there are now at most $arity(p) + 2$ patterns $P[x'_\ell, x'_{\ell+1}]$ below $P^+[t_1, t_2]$ in Q'. Indeed, if this were not the case, we could find a matching pair $i < j$ among the remaining patterns. Since $i_1 < j_1$ is maximal w.r.t. index $s + 1$, and there are no further matching pairs starting from j_m, we know that $i \geq i_1$ and $i < j_m$. Moreover, since a_i is still present in Q', it must be the case that $j_g < i < i_{g+1}$ for some $1 \leq g < m$. But this contradicts the fact that $i_{g+1} < j_{g+1}$ is maximal w.r.t. j_g.

In order for the different remaining patterns to form a sequence, we will need to perform a renaming of terms. If there are n patterns left under $P^+[t_1, t_2]$, we rename these patterns from left to right by:

$$P[t_1 = x''_0, x''_1], P[x''_1, x''_2], \ldots, P[x''_{n-1}, x''_n = t_2].$$

and rename the atoms underneath these patterns accordingly. We will also update π by setting $\pi(x''_i) = \pi(x'_j)$ if x'_j was renamed into x''_i and there is no $x'_{j'}$ with $j' < j$ that was also renamed into x''_i.

Let Q' be the instance obtained in this manner. We note that by construction, it is an instance of interest of $(\mathcal{Q}, \mathbb{P})$ w.r.t. R, as we only expand patterns into atoms that use the predicate p from the rule head, and the number of patterns generated from any repeatable pattern is at most $arity(p) + 2$.

Regarding π, note that a term may be shared among several repeatable patterns that are relevant for μ. However, we can show that if a term is shared by multiple relevant patterns, then the (partial) mapping associated with those patterns will agree on the shared term, i.e., π is well-defined. First note if a term is shared by two repeatable patterns, then it must appear as one of the distinguished terms (t_1, t_2) in both patterns. Moreover, by tracing the above construction, we find that π is the identity on such terms.

To complete the definition of π, we extend it to all of the terms of Q' by letting π be the identity on all terms that do not occur underneath a developed repeatable pattern (i.e., terms that appear in a repeatable pattern that is not expanded, or in one of the standard atoms of $\mathcal{Q}$). Observe that π is an injective function, so its inverse π^{-1} is well-defined.

Now let Q'_2 consist of all atoms in $Q' \cap Q_2$ that are not expanded from a repeatable pattern (i.e., they are standard atoms from $\mathcal{Q}$) as well as all atoms in Q' that lie under a repeatable pattern.

Note that by construction every term t in Q'_2 is such that $\pi(t)$ appears in Q_2. We can thus define a partition P'_u of the terms in $Q'_2 \cup H$ by taking every class C in P_u and replacing every term t from Q_2 by $\pi^{-1}(t)$, if such a term exists, and otherwise deleting t; terms from $p(\vec{t})$ are left untouched. Moreover, by the injectivity of π, every term appears in at most one class, i.e., P'_u is indeed a partition.

We aim to show that $\mu' = (Q'_2, p(\vec{t}), P'_u)$ is the desired unifier. We first show that μ' is a unifier of Q' with R. In what follows, it will prove convenient to extend π to the terms in the head atom $p(\vec{t})$, by letting π be the identity on such terms. We will let σ be a substitution associated with μ, and let σ' be the corresponding substitution for μ' defined by setting $\sigma'(t) = \sigma(\pi(t))$.

- P'_u is admissible: since π is the identity on constants, if a class in P'_u contains two constants c, d, then the corresponding class in P_u must also contain c, d (a contradiction).

- $\sigma'(p(\vec{t})) = \sigma'(Q'_2)$: since $\sigma'(p(\vec{t})) = \sigma(p(\vec{t}))$ (due to our choice of σ'), it suffices to prove that $\sigma'(Q'_2) \subseteq \sigma(Q_2)$. First take some atom α that belongs to $Q'_2 \cap Q_2$. Then we have $\pi(\alpha) = \alpha$, so $\sigma'(\alpha) = \sigma(\pi(\alpha)) \in \sigma(Q_2)$. Next consider the case of an atom α that belongs to Q_2 but not Q'_2. Then α must lie below a repeatable pattern $P^+[t_1, t_2]$ that is expanded into $k > arity(p) + 2$ standard patterns $P[t_1 = x_0, x_1], P[x_1, x_2], \ldots, P[x_{k-2}, x_{k-1}], P[x_{k-1}, x_k = t_2]$ in Q. In this case, $P^+[t_1, t_2]$ is expanded in Q' into

$$P[t_1 = x'_s, x'_{s+1}], P[x'_{s+1}, x'_{s+2}], \ldots, P[x'_{e-1}, x'_e = t_2],$$

  and each $P[x'_i, x'_{i+1}]$ is expanded into a'_i. If $e - s \leq arity(p) + 2$, then the atoms a'_i all belong to Q'_2. If we have $\alpha = a'_i$, then we have $\pi(a'_i) = a_i$, hence $\sigma'(\alpha) = \sigma(\pi(a'_i)) = \sigma(a_i) \in \sigma(Q_2)$. The final possibility is that $e - s > arity(p) + 2$, in which case some of the patterns will be removed and the remaining patterns will be renamed (as will be their corresponding atoms). Suppose that α is the atom a''_h below the pattern $P[x''_\ell, x''_{\ell+1}]$, which was obtained from renaming the pattern $P[x_h, x_{h+1}]$. We claim that $\sigma'(\alpha) = \sigma(\pi(\alpha)) = \sigma(a_h)$, hence $\sigma'(\alpha) \in \sigma(Q_2)$. By examining the way renaming is performed, there are two situations that can occur:

  - $\pi(x''_\ell) = x_h$ and $\pi(x''_{\ell+1}) = x_{h+1}$: in this case, $\pi(a''_h) = a_h$, hence $\sigma'(\alpha) = \sigma(a_h)$.
  - $\pi(x''_\ell) \neq x_h$: in this case, there must exist a matching pair $i_g < j_g$ such that $h = j_g$, $\pi(x''_\ell) = x_{i_g+1}$, and $\pi(x''_{\ell+1}) = x_{h+1}$. From the definition of matching pairs, we know that $\sigma(x_{i_g+1}) = \sigma(x_{j_g})$. It follows that $\sigma'(x''_\ell) = \sigma(\pi(x''_\ell)) = \sigma(x_{i_g+1}) = \sigma(x_h)$ and $\sigma'(x''_{\ell+1}) = \sigma(\pi(x''_{\ell+1})) = \sigma(x_{h+1})$. We can thus conclude that $\sigma'(\alpha) = \sigma(a_h)$.

- for a contradiction, suppose the class C' in P'_u contains an existential variable z from H and either a constant or a variable that occurs in $Q' \setminus Q'_2$. If it contains

a constant c, then the corresponding class C in P_u will contain both z and c, i.e., C is not a valid class. Next suppose that C' contains a variable x that occurs in $Q' \setminus Q'_2$, which means that the corresponding class C in P_u contains $\pi(x)$. Since x occurs in $Q' \setminus Q'_2$, it must either appear in a standard atom of Q' that does not appear under any repeatable pattern or in a repeatable pattern that is not developed in Q'. In the former case, the same atom appears in $Q \setminus Q_2$, and in the latter case, since Q is full, there is an atom in Q that is developed from the repeatable pattern and contains $\pi(x)$, but which does not participate in Q_2. In both cases, we obtain a contradiction.

Finally, we show that μ' is more general than μ:

- $\sigma'(Q'_2) \subseteq \sigma(Q_2)$: proven above.
- if $\sigma'(u_1) = \sigma'(u_2)$, then u_1, u_2 belong to the same class in P'_u, and so $\pi(u_1)$ and $\pi(u_2)$ must belong to the same class in P_u.

We have thus shown that Q' is an instance of interest of $(\mathcal{Q}, \mathbb{P})$ w.r.t. R such that there is a unifier μ' of Q' with R with $\mu' \geq \mu$. □

Rewriting with respect to internal unifiers is performed "inside" a repeatable pattern, independently from the other patterns and atoms in the query. We will therefore handle this kind of rewriting in a query-independent manner by updating the pattern definitions.

To find all internal unifiers of instances under a repeatable pattern $P^+[t_1, t_2]$ with a rule head $H = p(\vec{x})$, one may think that it is sufficient to consider each atom a_i in P's definition and check if there is an internal unifier of a_i with H. Indeed, this suffices when predicates are binary: in an internal unifier, t_1 and t_2 are unified with distinct variables, which cannot be existential; thus, the variables in H are frontier variables, and a piece necessarily consists of a single atom. However, if the arity of p is greater than 2, the other variables can be existential, so it may be possible to unify a path of atoms from P's definition, while it is not for a single atom. The next example illustrates this case.

Example 4.7
Let $R = s(x, y) \to r(z_1, x, z_2, y)$ be a rule and

$$P := r(\#2, \#1, x_0, x_1) | r(\#1, x_2, \#2, x_3) | r(x_4, x_5, \#1, \#2)$$

be a pattern definition. There is no internal unifier of an atom in P's definition with $H = r(z_1, x, z_2, y)$, since in all cases t_1 or t_2 is unified with an existential variable. However, if we expand $P^+[t_1, t_2]$ into a path $P[t_1, y_0], P[y_0, y_1], P[y_1, t_2]$, then expand each i^{th} pattern of this path into the i^{th} atom in P's definition, the resulting instance $r(y_0, t_1, x_0, x_1), r(y_0, x_2, y_1, x_3), r(x_4, x_5, y_1, t_2)$ can be unified with H by an internal unifier with the following partition:

$$\{\{z_1, y_0, x_4\}, \{x, t_1, x_2, x_5\}, \{z_2, x_0, y_1\}, \{y, x_1, x_3, t_2\}\}$$

Fortunately, we can bound the length of paths to be considered using both the arity of the predicate p and the number of atoms with predicate p in P's definition, allowing us to use the instances of interest defined earlier.

Definition 4.11 (Internal Direct Rewriting)
An (internal) direct rewriting $\mathbb{P}'$ *of a set of pattern definitions* $\mathbb{P}$ *w.r.t. a pattern name* P *and a rule* $R = (B, H) \in \mathcal{R}_L^+$ *is the set of pattern definitions obtained from* $\mathbb{P}$ *by updating* P*'s definition as follows. We consider the PCQ* $(\mathcal{Q} = P^+[x, y], \mathbb{P})$*. We select an instance of interest* Q *of* $\mathcal{Q}$ *w.r.t.* R*, an internal unifier* μ *of* Q *with* H*, and a substitution* σ *associated with* μ *that preserves the external terms. Let* B' *be obtained from* $\sigma(B)$ *by substituting the first (resp. second) external term by* $\#1$ *(resp.* $\#2$*). If* B' *is an atom, we add it to* P*'s definition. Otherwise,* B' *is a repeatable pattern of the form* $S^+[\#1, \#2]$ *or* $S^+[\#2, \#1]$*. Let* f *be the bijection on* $\{\#1, \#2\}$ *defined as follows: if* B' *is of the form* $S^+[\#1, \#2]$*,* f *is the identity, otherwise* f *permutes* $\#1$ *and* $\#2$*. For all atoms* s_i *in the definition of* S*, we add* $f(s_i)$ *to* P*'s definition.*

Note that the addition of an atom to a pattern definition is up to *isomorphism* (with #1 and #2 treated as distinguished variables, i.e., #1 and #2 are mapped to themselves).

The following example illustrates Step 3 of the algorithm.

Example 4.8
Assume $\mathcal{R}_L^+$ *contains the following rules:*

- $R_1 : P_2^+[x, y] \to s_1(x, y)$
- $R_2 : s_1(x, y) \to p_1(x, y)$
- $R_3 : s_2(x, y) \to p_2(y, x)$
- $R_4 : P_1^+[x, y] \to p_2(y, x)$

Note that the transitive predicates p_1 *and* p_2 *are recursive. Also, since rules contain no existential variable, each single-piece unifier unifies a single atom from the queries* $P_1^+[x, y]$ *and* $P_2^+[x, y]$*, therefore the only useful instances of interest have a single atom.*

Starting from $\mathbb{P} = \mathbb{P}_0$ *the following set of pattern definitions:*

- $P_1 := p_1(\#1, \#2)$
- $P_2 := p_2(\#1, \#2)$

the computations of Step 3 may, for instance, be decomposed as follows:

1. P_1 *is rewritten with* R_2 *and* P_2 *with* R_3 *obtaining the following definitions:*

- $P_1 := p_1(\#1, \#2) \mid s_1(\#1, \#2)$
- $P_2 := p_2(\#1, \#2) \mid s_2(\#2, \#1)$

2. *P_1 is rewritten with R_1, which leads to "copy" the definition of P_2 into that of P_1:*

   - $P_1 := p_1(\#1, \#2) \mid s_1(\#1, \#2) \mid p_2(\#1, \#2) \mid s_2(\#2, \#1)$
   - $P_2 := p_2(\#1, \#2) \mid s_2(\#2, \#1)$

3. *P_2 is rewritten with R_4 which leads to copy the definition of P_1 into that of P_2 (with f permuting $\#1$ and $\#2$):*

   - $P_1 := p_1(\#1, \#2) \mid s_1(\#1, \#2) \mid p_2(\#1, \#2) \mid s_2(\#2, \#1)$
   - $P_2 := p_2(\#1, \#2) \mid s_2(\#2, \#1) \mid p_1(\#2, \#1) \mid s_1(\#2, \#1) \mid p_2(\#2, \#1) \mid s_2(\#1, \#2)$

4. *P_1 can be enriched again, which leads in turn to enrich P_2 and we finally obtain the same definitions for P_1 and P_2:*

   - $P_1 := p_1(\#1, \#2) \mid s_1(\#1, \#2) \mid p_2(\#1, \#2) \mid s_2(\#2, \#1) \mid p_1(\#2, \#1) \mid s_1(\#2, \#1) \mid p_2(\#2, \#1) \mid s_2(\#1, \#2)$
   - $P_2 = P_1$

Proposition 4.4

Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ where a repeatable pattern $P^+[t_1, t_2]$ occurs and $R \in \mathcal{R}_L^+$. For any instance Q of $(\mathcal{Q}, \mathbb{P})$, any classical direct rewriting $\mathcal{Q}'$ of Q with R w.r.t. a unifier internal to $P^+[t_1, t_2]$, and any $Q' \in full(\mathcal{Q}', \mathbb{P})$, there exists a direct rewriting $\mathbb{P}'$ of $\mathbb{P}$ w.r.t. P and R such that $(\mathcal{Q}, \mathbb{P}')$ has a full instance that is isomorphic to Q'.

Proof: Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ where $P^+[t_1, t_2]$ occurs, $R = (B, H) \in R_L^+$, Q be an instance of $(\mathcal{Q}, \mathbb{P})$, $\mu = (Q_2, H, P_u)$ be a unifier internal to $P^+[t_1, t_2]$ of Q with R, $\mathcal{Q}'$ be the classical direct rewriting of Q with R w.r.t. μ, and Q' be a full instance of $(\mathcal{Q}', \mathbb{P})$.

Since μ is internal to $P^+[t_1, t_2]$, all atoms in Q_2 are expanded from $P^+[t_1, t_2]$ in Q, and do not unify t_1 with t_2, nor t_1 (resp. t_2) with an existential variable from H. We denote by $P[t_1 = x_0, x_1], P[x_1, x_2], \ldots, P[x_{k-1}, x_k = t_2]$ the sequence of standard patterns expanded under $P^+[t_1, t_2]$ in Q, x_s and x_e $(s < e)$ the external terms of $P^+[t_1, t_2]$ w.r.t. μ, and a_i the atom expanded under $P[x_i, x_{i+1}]$. From Proposition 4.3, there is a unifier μ' of an instance of interest Q_3 of $\mathcal{Q}$ with R with $\mu' \geq \mu$. Since x_s and x_e are not unified with existential variables, let Q_4 be the CQ obtained from

Q_3 by removing all atoms and patterns that are not relevant for μ'. Obviously Q_4 is an instance of interest of a PCQ of the form $P^+[t_1, t_2]$. Let $\mathbb{P}'$ be the direct rewriting of $\mathbb{P}$ w.r.t. μ', obtained from Q_4.

Let us consider the following notations:

- $A_l = \{P[x_i, x_{i+1}] \mid 0 \leq i < s\}$,
- $A_m = \{P[x_i, x_{i+1}] \mid s \leq i < e\}$,
- $A_r = \{P[x_i, x_{i+1}] \mid e \leq i < k\}$,
- $A = A_l \cup A_m \cup A_r$.

Furthermore let A'_l (resp. A'_m, A'_r, A') be the set of atoms expanded under A_l (resp. A_m, A_r, A) in Q.

Initialize Q'' to $Q \setminus A' \cup \{P^+[t_1, t_2]\}$. One can see that Q'' is an instance of both $(\mathcal{Q}, \mathbb{P})$ and $(\mathcal{Q}, \mathbb{P}')$. If B is a not a repeatable pattern, let $\ell = 1$, otherwise let S be the name of the repeatable pattern in B, and $S[x'_0, x'_1], \ldots, S[x'_{\ell-1}, x'_\ell]$ be the sequence expanded from $S^+[x'_0, x'_\ell]$ in Q'. We denote by a'_i the atom expanded under $S[x'_i, x'_{i+1}]$. Then expand $P^+[t_1, t_2]$ in Q'' into $k' = |A_l| + |A_r| + \ell$ standard patterns: $P[t_1 = x''_0, x''_1], \ldots, P[x''_{k-1}, x''_{k'} = t_2]$. Let π be the function defined as follows:

- for all $0 \leq i \leq s$, $\pi(x''_i) = x_i$;
- for all $s < i < s + \ell$, $\pi(x''_i) = x'_{i-s}$;
- for all $s + \ell \leq i \leq k'$, $\pi(x''_i) = x_{i-\ell+(e-s)}$.

Note that π is injective, so its inverse is also a function. Expand all $P[x''_i, x''_{i+1}]$ with $0 \leq i < s$ or $s + \ell \leq i < k$ (resp. $s \leq i < s + \ell$) into $\pi^{-1}(a_i)$ (resp. $\pi^{-1}(a'_i)$). Finally, for all terms u in Q'' for which π is not defined (i.e., those terms appearing in atoms that were not expanded from the pattern $P^+[t_1, t_2]$), we set $\pi(u) = u$.

By construction, Q'' is still an instance of $(\mathcal{Q}, \mathbb{P}')$ and π is an isomorphism from Q'' to Q'. □

Now that internal direct rewritings are covered, we detail how we handle the case of external direct rewritings. Given a PCQ and a rule, the idea is to consider all instances of interest of the PCQ w.r.t. the rule and all associated external unifiers. This is sufficient to "cover" (in the sense of Proposition 4.3) all external unifiers with respect the rule. However, to be complete, we have to keep in mind that a repeatable pattern $P^+[t_1, t_2]$ can be expanded into any number of atoms in full instances. Hence, each repeatable pattern $P^+[t_1, t_2]$ in an instance of interest is actually seen as a sequence $P^+[t_1, x_1], X[x_1, x_2], P^+[x_2, t_2]$ where $X[x_1, x_2]$ is the sequence of atoms (from the subtree of $P^+[t_1, t_2]$) involved in the unifier, and each of the surrounding P^+ pattern may be present or not, which yields four cases for each pattern. Moreover, we have to check that each of these cases (which will at most be four to the power of the number of repeatable patterns in the query) is actually possible, i.e., that the unifier still satisfies the conditions for being a unifier.

Definition 4.12 (External Direct Rewriting)
Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ, $R \in \mathcal{R}_L^+$, T be an instantiation of interest of $(\mathcal{Q}, \mathbb{P})$ w.r.t. R, Q be the instance associated with T, and $\mu = (Q', H, P_\mu)$ be an external unifier of Q with R. From this, several direct rewritings of $\mathcal{Q}$ w.r.t. $\mathbb{P}$ and R can be built. First, we mark all leaves in T that are labelled by an atom in Q', and we restrict T to branches leading to a marked leaf. Then, we consider each instantiation T_i that can be obtained from $\mathcal{Q}$ as follows. Replace each repeatable pattern $P^+[t_1, t_2]$ that has $k > 0$ children in T by one of the following paths (using fresh variables x_i):

(i) $P^+[t_1, x_1] \wedge X[x_1, x_2] \wedge P^+[x_2, t_2]$,

(ii) $P^+[t_1, x_1] \wedge X[x_1, t_2]$,

(iii) $X[t_1, x_2] \wedge P^+[x_2, t_2]$,

(iv) $X[t_1, t_2]$,

where $X[v_1, v_2]$ is a sequence of k standard patterns $P[v_1, y_1], P[y_1, y_2], \ldots, P[y_{k-1}, v_2]$ (the y_i being fresh variables). Let $\mathcal{Q}_i$ be the instance associated with T_i.

If $P[x', y']$ in T has child $a(\vec{t})$, expand in T_i the corresponding $P[x, y]$ into $a(\rho(\vec{t}))$ where $\rho = \{x' \mapsto x, y' \mapsto y\}$. If $\mu' = (\rho(Q'), H, \rho(P_\mu))$ is still a unifier of $\mathcal{Q}_i$ with H, we say that $\mathcal{Q}_i$ is a minimally-unifiable instance *of $\mathcal{Q}$ w.r.t. μ. In this case, $\mathcal{Q}'_i = \mu'(Q_i) \setminus \mu'(H) \cup \mu'(B)$ is an* (external) direct rewriting *of $\mathcal{Q}$ w.r.t. $\mathbb{P}$ and R.*

Example 4.9
Consider again $R_2 = s_1(x', y') \rightarrow s_2(x', y', z')$ and

$$Q_3 = s_2(a, y_0, z) \wedge s_2(x_1, y_1, z) \wedge s_2(b, y_2, x_1) \wedge s_1(a, b)$$

from Examples 4.5 and 4.6 respectively, and let

$$\mu = (\{s_2(a, y_0, z), s_2(x_1, y_1, z)\}, H_2, \{\{a, x_1, x'\}, \{y_0, y_1, y'\}, \{z, z'\}\})$$

Figure 4.4 depicts the instantiation of the PCQ leading to Q_3.

First, we consider the instantiation that generated Q_3, and we remove the vertex labelled by $P_2[x_1, b]$ and its child $s_2(b, y_2, x_1)$, since the latter atom is not involved in μ. Next we replace the repeatable pattern $P_1^+[a, z]$ (resp. $P_2^+[z, b]$) using one of the four cases detailed above, and we check whether μ' (obtained from μ) is still a unifier. Concerning $P_1^+[a, z]$, the four following cases are to be considered:

(i) $P_1^+[a, t_1], s_2(t_1, y_0, t_2), P_1^+[t_2, z]$

(ii) $P_1^+[a, t_1], s_2(t_1, y_0, z)$

(iii) $s_2(a, y_0, t_2), P_1^+[t_2, z]$

(iv) $s_2(a, y_0, z)$

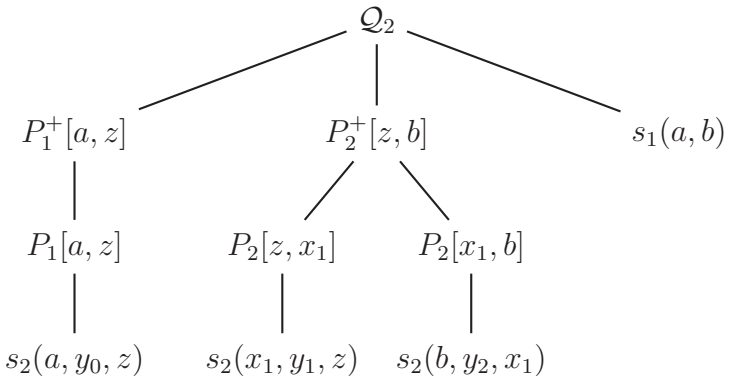

Figure 4.4: Instantiation of Q_3 from Example 4.9

In case (i), t_2 is unified with the existential variable z', hence $P_1^+[t_2, z]$ should be unified as well, which is not possible. The same holds for case (iii). Finally only cases (ii) and (iv) are suitable to replace $P_1^+[a, z]$.

Concerning $P_2^+[z, b]$, the four following cases are to be considered:

(i) $P_2^+[z, v_1], s_2(v_2, y_1, v_1), P_1^+[v_2, b]$

(ii) $P_2^+[z, v_1], s_2(b, y_1, v_1)$

(iii) $s_2(v_2, y_1, z), P_1^+[v_2, b]$

(iv) $s_2(b, y_1, z)$

Case (i) and (ii) are excluded for the same reason as above (v_1 is unified with z').

Now let us consider the PCQs obtained by combining all considered possible expansions of P_1^+ and P_2^+:

- $\mathcal{Q}_1 = P_1^+[a, t_1] \wedge\ s_2(t_1, y_0, z) \wedge\ s_2(v_2, y_1, z)\ \wedge\ P_2^+[v_2, b]\ \wedge\ s_1(a, b)$
- $\mathcal{Q}_2 = P_1^+[a, t_1] \wedge\ s_2(t_1, y_0, z) \wedge\ s_2(b, y_1, z)\ \wedge\ s_1(a, b)$
- $\mathcal{Q}_3 = s_2(a, y_0, z) \wedge\ s_2(v_2, y_1, z)\ \wedge\ P_2^+[v_2, b]\ \wedge s_1(a, b)$
- $\mathcal{Q}_4 = s_2(a, y_0, z) \wedge\ s_2(b, y_1, z)\ \wedge\ s_1(a, b)$

We observe that no unifier exists when we choose case (iv) for both patterns, indeed $\{s_2(a, y_0, z), s_2(b, y_1, z)\}$ cannot be unified with $s_2(x', y', z')$ since it would mean to unify constants a and b together.

By considering the three minimally-unifiable instances $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ we obtain the following rewritings:

- $\mathcal{Q}'_1 = P_1^+[a, x'] \wedge s_1(x', y') \wedge P_2^+[x', b] \wedge s_1(a, b)$,

- $\mathcal{Q}_2 = P_1^+[a,b] \wedge\ s_1(b,y') \wedge\ \wedge\ s_1(a,b)$
- $\mathcal{Q}'_3 = s_1(a,y') \wedge\ P_2^+[a,b] \wedge\ s_1(a,b)$

Proposition 4.5

Let $(\mathcal{Q},\mathbb{P})$ be a PCQ and $R \in \mathcal{R}_L^+$. For every $Q \in full(\mathcal{Q},\mathbb{P})$ and every classical direct rewriting Q' of Q with R w.r.t. an external unifier, there is a direct rewriting $\mathcal{Q}'$ of $\mathcal{Q}$ w.r.t. $\mathbb{P}$ and R that has an instance isomorphic to Q'.

Proof: Let $(\mathcal{Q},\mathbb{P})$ be a PCQ, $R = (B,H) \in R_L^+$, $Q \in full(\mathcal{Q},\mathbb{P})$, $\mu = (Q_u, H, P_u)$ be an external unifier of Q with R, and Q' be the classical direct rewriting of Q with R w.r.t. μ.

From Proposition 4.3, there is an instance of interest Q_2 of $(\mathcal{Q},\mathbb{P})$ such that there is a unifier $\mu' = (Q_{u'}, H, P_{u'}) \geq \mu$ of Q_2 with R. We denote by σ (resp. σ') a substitution associated with μ (resp. μ').

For any repeatable pattern $P^+[t_1,t_2]$ in $\mathcal{Q}$, build A, A_l, A_m and A_r as in the proof of Proposition 4.4 using the instance Q_2 and unifier μ'. Assume t_1 (or t_2) is unified with an existential variable, then from the condition on external unifiers, A_l (or A_r) is empty. Consider the minimally-unifiable instance $\mathcal{Q}_M$ of $\mathcal{Q}$ w.r.t. μ' that replaces $P^+[t_1,t_2]$ by: (i) A_m if $A_l = A_r = \emptyset$; (ii) $P^+[t_1,x_s], A_m$ if $A_r = \emptyset$ and $A_l \neq \emptyset$; or (iii) $A_m, P^+[x_e,t_2]$ if $A_l = \emptyset$ and $A_r \neq \emptyset$. In case (ii) (resp. (iii)), since no atom in A_l (resp. A_r) is involved in μ', x_s (resp. x_e) is not unified with an existential variable (or the piece condition on unifiers would not be satisfied). Therefore, μ' is a unifier of $\mathcal{Q}_M$ with R. We let $\mathcal{Q}'$ be the direct rewriting of $\mathcal{Q}_M$ w.r.t. μ' and R.

Note that each repeatable pattern $P^+[t_1,t_2]$ in Q' expands into $A_l \wedge \sigma(B) \wedge A_r$, and in $\mathcal{Q}'$ there is a $P^+[t_1,x_s]$ (resp. $P^+[x_e,t_2]$) iff A_l (resp. A_r) is not empty. Thus consider Q'' obtained from $\mathcal{Q}'$ by expanding $P^+[t_1,x_s]$ (resp. $P^+[x_e,t_2]$) into k standard patterns where $k = |A_l|$ (resp. $k = |A_r|$), and choose the same atoms as in A'_l (resp. A'_r). Since $\mu' \geq \mu$, there is a homomorphism π from $\sigma'(Q_{u'})$ to $\sigma(Q_u)$. Note that if we restrict the domain of π to terms in $\sigma'(B)$, π is an isomorphism. Furthermore, we can extend the domain ofπ to Q'' in the same way as we did in the previous proof. Thus Q'' is isomorphic to Q'. □

4.2.4 Termination and Correctness

Our algorithm, while not being guaranteed to terminate, is correct: indeed, if the query Q is entailed by the knowledge base then the algorithm produces a set of Datalog rules $\Pi_{\mathbb{P}}$ (always finite) and a CQ Q' (from a possibly infinite set $\mathcal{Q}_{\mathbb{Q}}$) such that the evaluation of $\Pi_{\mathbb{P}} \cup \{Q'\}$ (as a Datalog query) on the set of facts yields a positive answer, and reciprocally, as precisely stated by the next theorem.

Theorem 4.6

Let Q be a CQ, $(\mathcal{F}, \mathcal{R})$ be a *linear+trans* knowledge base, and $(\Pi_{\mathbb{P}}, Q_{\mathbb{Q}})$ be the (possibly infinite) output of the algorithm; then:

$$(\mathcal{F}, \mathcal{R}) \models Q \Leftrightarrow \exists Q' \in Q_{\mathbb{Q}} : (\mathcal{F}, \Pi_{\mathbb{P}}) \models Q'$$

The proof of this theorem is quite technical, and thus is split into the following five lemmas.

Lemma 4.1

Let Q be a CQ, $R \in \mathcal{R}_T$, $\mathbb{P}_0$ be the initial set of pattern definitions relative to $\mathcal{R}^T$ (see Step 1 of the algorithm overview), and $\mathcal{Q}^+$ be obtained from Q by replacing all atoms $p(t_1, t_2)$ such that p is a transitive predicate by $P^+[t_1, t_2]$. If there is a classical direct rewriting Q' of Q with R, then there is a full instance Q'' of $(\mathcal{Q}^+, \mathbb{P}_0)$ that is isomorphic to Q'.

Proof: Let $p(t_1, t_2)$ be the atom of Q that is rewritten to obtain Q'. Since p is a transitive predicate, it occurs in a pattern definition P_0 in $\mathbb{P}$, and $\mathcal{Q}^+$ contains the atom $P^+[t_1, t_2]$. In Q', $p(t_1, t_2)$ is rewritten into $p(t_1, x_1) \wedge p(x_1, t_2)$. Let Q'' be the full instance of $(\mathcal{Q}^+, \mathbb{P}_0)$ that expands all repeatable patterns but $P^+[t_1, t_2]$ into a single standard pattern, expands $P^+[t_1, t_2]$ into two standard patterns $P[t_1, x'_1], P[x'_1, t_2]$, and then further expands the standard patterns using the unique atom in each of the pattern definitions. It is clear that Q'' is isomorphic to Q' (simply map x'_1 to x_1 and all other terms to themselves). □

Lemma 4.2

Let $\mathbb{P}$ be a set of pattern definitions, $\mathbb{P}_0 \subseteq \mathbb{P}$ be the initial set of pattern definitions built from the set $\mathcal{R}_T$ of transitivity rules, $(\mathcal{Q}, \mathbb{P})$ be a PCQ that does not contain any standard atom using a transitive predicate, Q be a full instance of $(\mathcal{Q}, \mathbb{P})$, and $\mathcal{Q}^+$ be obtained from Q by replacing all atoms $p(t_1, t_2)$ with p transitive by $P^+[t_1, t_2]$.
Then, for every full instance Q' of $(\mathcal{Q}^+, \mathbb{P}_0)$, there is a full instance Q'' of $(\mathcal{Q}, \mathbb{P})$ such that Q'' is isomorphic to Q'.

Proof: We build the instance Q'' as follows. Initialize Q'' to the atoms and repeatable patterns occurring in $\mathcal{Q}$. Next, for all repeatable patterns $P_i^+[t_1, t_2]$ in the

instantiation underlying Q consider each of the atom that is expanded from a child of $P_i^+[t_1, t_2]$ in turn, working from left to right. If the atom $p(\vec{t})$ under $P_i[u, v]$ is being considered, then do the following:

- if p is not a transitive predicate, then add a single child $P_i[u, v]$ to $P_i^+[t_1, t_2]$, and expand it into $p(\vec{t})$.
- if p is transitive, then $p(\vec{t})$ has been replaced in $\mathcal{Q}^+$ by $P^+[\vec{t}]$. We also know that $\vec{t}$ consists of the terms u, v from $P_i[u, v]$. We suppose that $p(\vec{t}) = p(u, v)$ (hence $P^+[\vec{t}] = P^+[u, v]$); a similar argument can be used if the positions are reversed. Let $P[u = x_0, x_1], \ldots, P[x_{k-1}, x_k = v]$ be the children of $P^+[u, v]$ in Q', and a_ℓ be the atom expanded under $P[x_\ell, x_{\ell+1}]$ $(0 \leq \ell < k)$. In place of the child $P_i^+[u, v]$ in $\mathcal{Q}^+$, we will add k children to $P_i^+[t_1, t_2]$ in Q'': $P_i[u = x_0, x_1], \ldots, P_i[x_{k-1}, x_k = v]$, and expand $P_i[x_j, x_{j+1}]$ into a_j. Note that we may assume that the terms x_i $(0 < i < k)$ are fresh, i.e., they do not already appear in Q''.

It can be verified that the resulting full instance Q'' is isomorphic to Q'. Indeed, all atoms in Q' that are also in Q are present in Q''. All other atoms belong to a sequence of transitive atoms, which we have reproduced (up to renaming of variables) in Q''. □

Lemma 4.3

Let Q be a CQ and $\mathcal{R}$ be a set of *linear+trans* rules, and let $(\mathbb{Q}, \mathbb{P})$ be the output of the algorithm. For any Q' obtained from a sequence of classical direct rewritings of Q with $\mathcal{R}$, there is a PCQ $(\mathcal{Q}, \mathbb{P})$ with $\mathcal{Q} \in \mathbb{Q}$ and a full instance Q'' of $(\mathcal{Q}, \mathbb{P})$ such that Q'' is isomorphic to Q'.

Proof: Let $Q = Q_0, \mu_1, Q_1, \mu_2, Q_2, \ldots, \mu_k, Q_k = Q'$ be a sequence of classical direct rewritings from Q to Q', and let $R_1, \ldots, R_k$ be the associated sequence of rules from $\mathcal{R}$.

We show the desired property, by induction on $0 \leq i \leq k$. For the base case $(i = 0)$, we can set $\mathcal{Q}_0 = Q^+$, since $Q_0 = Q$ is clearly a full instance of $(\mathcal{Q}_0, \mathbb{P})$.

For the induction step, suppose that we have $\mathcal{Q}_{i-1} \in \mathbb{Q}$ and a full instance Q''_{i-1} of $(\mathcal{Q}_{i-1}, \mathbb{P})$ that is isomorphic to the CQ Q_{i-1}. There are two cases to consider, depending on the type of the rule R_i.

If R_i is a transitivity rule, then from Lemma 4.1, we know that $\mathcal{Q}^+_{i-1}$ (obtained from Q_{i-1} by replacing every transitive predicate p by pattern P^+) is such that there is a full instance Q^+_{i-1} of $(\mathcal{Q}^+_{i-1}, \mathbb{P})$ that is isomorphic to Q_i. Furthermore, we know that $\mathcal{Q}_{i-1}$ cannot contain any standard atom with a transitive predicate, since every PCQ produced in Step 5 contains patterns for the transitive predicates. Thus, we

may apply Lemma 4.2 and infer that Q^{+}_{i-1} is isomorphic to some full instance Q''_{i-1} of $(\mathcal{Q}_{i-1}, \mathbb{P})$. Therefore, Q''_{i-1} is isomorphic to Q_i.

If R_i is not a transitive rule, since Q_{i-1} is isomorphic to some full instance Q''_{i-1} of $(\mathcal{Q}_{i-1}, \mathbb{P})$, let μ''_i be the unifier of Q''_{i-1} with R_i obtained from μ_i and the isomorphism between Q_{i-1} and Q''_{i-1}. If μ''_i is internal to some repeatable pattern, then from Proposition 4.4, we know that there is an instance Q'_i of $(\mathcal{Q}_{i-1}, \mathbb{P})$ that is isomorphic to Q_i. Otherwise, from Proposition 4.5, there exist μ'_i and a direct rewriting $\mathcal{Q}_i$ of $\mathcal{Q}_{i-1}$ with μ'_i such that there is an instance Q'_i of $(\mathcal{Q}_i, \mathbb{P})$ that is isomorphic to Q_i.

We have thus completed the inductive argument and can conclude that there is a PCQ $(\mathcal{Q}, \mathbb{P})$ with $\mathcal{Q} \in \mathbb{Q}$ and a full instance Q'' of $(\mathcal{Q}, \mathbb{P})$ such that Q'' is isomorphic to $Q' = Q_k$. □

Lemma 4.4

Let Q be a CQ, $(\mathcal{F}, \mathcal{R})$ be a *linear+trans* KB, and $(\Pi_{\mathbb{P}}, Q_{\mathbb{Q}})$ be the output of the algorithm. If $(\mathcal{F}, \mathcal{R}) \models Q$ then $(\mathcal{F}, \Pi_{\mathbb{P}}) \models Q'$ for some $Q' \in Q_{\mathbb{Q}}$.

Proof: Since $(\mathcal{F}, \mathcal{R}) \models Q$, there is a (finite) classical rewriting Q' of Q with $\mathcal{R}$ such that $\mathcal{F} \models Q'$. From Lemma 4.3, there is a PCQ $(\mathcal{Q}, \mathbb{P})$ with $\mathcal{Q} \in \mathbb{Q}$ and a full instance Q'' of $(\mathcal{Q}, \mathbb{P})$ such that Q'' is isomorphic to Q'. Therefore, $\mathcal{F} \models Q''$. We conclude by Proposition 4.2. □

Lemma 4.5

Let Q be a CQ, $(\mathcal{F}, \mathcal{R})$ be a *linear+trans* KB, and $(\Pi_{\mathbb{P}}, Q_{\mathbb{Q}})$ be the output of the algorithm. If $(\mathcal{F}, \Pi_{\mathbb{P}}) \models Q_{\mathbb{Q}}$ then $(\mathcal{F}, \mathcal{R}) \models Q$.

Proof: Let $\mathbb{P}$ be the set of pattern definitions computed in Step 3 of the algorithm, and $\Pi_{\mathbb{P}}$ the corresponding set of Datalog rules. Consider the CQ Q^{++}obtained from Q by replacing every atom $p(t_1, t_2)$ such that p is transitive by the atom $p^{+}(t_1, t_2)$. The following claim establishes the soundness of the internal rewriting mechanism in Step 3:

Claim 4.1 *If* $\mathcal{F}, \Pi_{\mathbb{P}} \models Q^{++}$, *then* $\mathcal{F}, \mathcal{R} \models Q$.

Proof of claim. Let $\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_k = \mathbb{P}$ be the sequence of sets of pattern definitions that led to $\mathbb{P}$ in Step 3, with $\mathbb{P}_{i+1}$ being obtained from $\mathbb{P}_i$ by a single direct (internal) rewriting step. We prove by induction two distinct properties expressed at rank $0 \leq j \leq k$:

P1 every rule in $\Pi_{\mathbb{P}_j}$ is a semantic consequence of $\Pi_{\mathbb{P}_0} \cup \mathcal{R}$.

P2 for every set of facts $\mathcal{F}'$ and CQ Q' (over the original vocabulary):

$$\mathcal{F}', \Pi_{\mathbb{P}_j} \models (Q')^{++} \quad \Rightarrow \quad \mathcal{F}', \mathcal{R} \models Q'$$

In the second property, $(Q')^{++}$ denotes the CQ obtained by replacing every atom $p(t_1, t_2)$ such that p is transitive by the atom $p^+(t_1, t_2)$. Observe that **P2** at rank k yields the claim: we simply take $\mathcal{F}' = \mathcal{F}$ and $Q' = Q$.

Base case ($i = 0$): property **P1** is obviously verified. For property **P2**, we note that $\mathbb{P}_0$ consists of the following rules for every transitive predicate p: the transitivity rule $p^+(x, y) \wedge p^+(y, z) \rightarrow p^+(x, z)$ and the initialization rule $p(x, y) \rightarrow p^+(x, y)$. Clearly, if $\mathcal{F}, \Pi_{\mathbb{P}_0} \models (Q')^{++}$, then we have $\mathcal{F}, \mathcal{R} \models Q'$, since if we can derive $p^+(a, b)$ using $\mathcal{F}, \Pi_{\mathbb{P}_0}$, then we can also derive $p(a, b)$ from $\mathcal{F}, \mathcal{R}$ using the transitivity rule for p in $\mathcal{R}$.

*Induction step for **P1**:* we assume property **P1** holds for some rank $0 \leq i < k$ and show that it holds also for $i + 1$.

Suppose that $\mathbb{P}_{i+1}$ is obtained from $\mathbb{P}_i$ by a single direct rewriting step w.r.t. pattern name P and the rule $R = B \rightarrow H \in \mathcal{R}_L^+$. Let $\mathcal{Q} = P^+[x, y]$, Q be the considered instance of interest of $\mathcal{Q}$ w.r.t. R, $\mu = (Q', H, P_u)$ be the considered internal unifier of Q with H, and σ be the considered substitution associated with μ that preserves the external terms. Finally, let B' be obtained from $\sigma(B)$ by substituting the first (resp. second) external term by #1 (resp. #2).

Since we know that μ is an internal unifier, the external terms of Q' cannot be unified together or with an existential variable. Thus by considering Q'' and $P_{u'}$ obtained from Q' and P_u by substituting the first (resp. second) external term by #1 (resp. #2), it is clear that $\mu' = (Q'', H, P_{u'})$ is a unifier of Q'' with R such that $\sigma'(B) = B'$, where σ' is the substitution associated with μ' that preserves the special terms #1 and #2.

We consider two cases depending on the nature of B'.

Case 1: The first possibility is that B' is an atom (as opposed to a repeatable pattern), in which case we add the following rule to $\Pi_{\mathbb{P}_i}$: $B' \rightarrow p^+(\#1, \#2)$.

Let $a_1, \ldots, a_k$ be the atoms of Q'', and let a'_j be the atom in P's definition from which a_j is obtained. Since there is a rewriting of $\{a_j \mid 0 < j \leq k\}$ with R into B' (using the unifier μ'), and the rule R appears in the original set of rules $\mathcal{R}$, it follows that

$$\mathcal{R} \models B' \rightarrow a_1 \wedge \cdots \wedge a_k$$

From the induction hypothesis, we know that the rules $a'_j \rightarrow p^+(\#1, \#2)$ $(0 < j \leq k)$ are entailed by $\Pi_{\mathbb{P}_0}, \mathcal{R}$. We also know that for all $1 < j \leq k$, the atoms a_{j-1} and a_j share a variable corresponding respectively to #2 in a'_{j-1} and to #1 in a'_j. Thus, by applying the rules $a'_j \rightarrow p^+(\#1, \#2)$ $(0 < j \leq k)$ to the conjunction $a_1 \wedge \cdots \wedge a_k$, we

obtain $p^+(\#1, x_1) \wedge p^+(x_1, x_2) \wedge \cdots \wedge p^+(x_{k-1}, \#2)$. Hence:

$$\Pi_{\mathbb{P}_0}, \mathcal{R} \models \bigwedge_{j=0}^{k} a_j \rightarrow p^+(\#1, x_1) \wedge p^+(x_1, x_2) \wedge \cdots \wedge p^+(x_{k-1}, \#2)$$

Since $\Pi_{\mathbb{P}_0}$ contains a transitivity rule for p^+, we can further infer that

$$\Pi_{\mathbb{P}_0} \models p^+(x_1, x_2) \wedge \cdots \wedge p^+(x_{k-1}, \#2) \rightarrow p^+(\#1, \#2)$$

By chaining together the preceding entailments, we obtain $\Pi_{\mathbb{P}_0}, \mathcal{R} \models (B' \rightarrow p^+(\#1, \#2))$, as desired.

Case 2: The other possibility is that B' is a repeatable pattern of the form $S^+[\#1, \#2]$ or $S^+[\#2, \#1]$. Let f be a bijection on $\{\#1, \#2\}$: if B' is of the form $S^+[\#1, \#2]$, f is the identity, otherwise f permutes $\#1$ and $\#2$. Then for all s_ℓ in the definition of S, we add $f(s_\ell)$ to P's definition, and we add the corresponding rules $f(s_\ell) \rightarrow p^+(\#1, \#2)$ to $\Pi_{\mathbb{P}_i}$. Consider one such rule rule $f(s_\ell) \rightarrow p^+(\#1, \#2)$.

Let $a_1, \ldots, a_k$ and $a'_1, \ldots, a'_k$ be defined as in Case 1. Since there is a rewriting of $\{a_j \mid 0 < j \leq k\}$ with $R \in \mathcal{R}_L^+$ into B', and since the rule R was obtained from a rule R' in $\mathcal{R}$ by replacing the transitive predicate s in the rule head by the repeatable pattern S^+, it follows that

$$\mathcal{R} \models f(s(\#1, \#2)) \rightarrow a_1 \wedge \cdots \wedge a_k$$

Arguing as in Case 1, we obtain

$$\Pi_{\mathbb{P}_0}, \mathcal{R} \models f(s(\#1, \#2)) \rightarrow p^+(\#1, \#2)$$

From the induction hypothesis, we know that that the rules $s_\ell \rightarrow s^+(\#1, \#2)$ are entailed from $\Pi_{\mathbb{P}_0}, \mathcal{R}$, and the same obviously holds for the rules $f(s_\ell) \rightarrow f(s^+(\#1, \#2))$. By combining the preceding entailments, we obtain $\Pi_{\mathbb{P}_0}, \mathcal{R} \models f(s_\ell) \rightarrow p^+(\#1, \#2)$.

*Induction step for property **P2**:* we assume **P2** holds for some rank $0 \leq i < k$ and show that it holds also for $i + 1$.

Suppose now that $\mathcal{F}', \Pi_{\mathbb{P}_{i+1}} \models (Q')^{++}$, for some set of facts $\mathcal{F}'$ and CQ Q' (over the original predicates). This means that there is a finite derivation sequence $\mathcal{F}' = \mathcal{F}_0^{++}, \ldots, \mathcal{F}_m^{++}$ such that $\mathcal{F}_m^{++} \models (Q')^{++}$ and such that for all $0 \leq \ell < m$, $\mathcal{F}_{\ell+1}^{++}$ is obtained from $\mathcal{F}_\ell^{++}$ either (i) by a sequence of applications of rules from $\Pi_{\mathbb{P}_i}$ or (ii) by a sequence of applications of rules from $\Pi_{\mathbb{P}_{i+1}} \setminus \Pi_{\mathbb{P}_i}$.

In case (i), we have $\mathcal{F}_\ell^{++}, \Pi_{\mathbb{P}_i} \models \mathcal{F}_{\ell+1^{++}}$. Letting $\mathcal{F}_r$ be the set of facts obtained by replacing every predicate p^+ in $\mathcal{F}_r^{++}$ by the corresponding predicate p, and recalling that $\Pi_{\mathbb{P}_i}$ contains the rule $p(x, y) \rightarrow p^+(x, y)$, we have $\mathcal{F}_\ell, \Pi_{\mathbb{P}_i} \models \mathcal{F}_{\ell+1}^{++}$. Applying the induction hypothesis (treating $\mathcal{F}_{\ell+1}^{++}$ as a CQ), we obtain $\mathcal{F}_\ell, \mathcal{R} \models \mathcal{F}_{\ell+1}$.

In case (ii), we have $\mathcal{F}_\ell^{++}, (\Pi_{\mathbb{P}_{i+1}} \setminus \Pi_{\mathbb{P}_i}) \models \mathcal{F}_{\ell+1}^{++}$. From property **P1**, we obtain $\mathcal{F}_\ell^{++}, \Pi_{\mathbb{P}_0}, \mathcal{R} \models \mathcal{F}_{\ell+1}^{++}$. Using the rule $p(x, y) \rightarrow p^+(x, y)$ (that is present in $\Pi_{\mathbb{P}_0}$), the

latter yields $\mathcal{F}_\ell, \Pi_{\mathbb{P}_0}, \mathcal{R} \models \mathcal{F}_{\ell+1}^{++}$. Finally, we note that if we can derive $p^+(a,b)$ from $\mathcal{F}_\ell, \Pi_{\mathbb{P}_0}, \mathcal{R}$, then we can also infer $p(a,b)$ from $\mathcal{F}_\ell, \mathcal{R}$ by using the transitivity rule for p instead of using $p(x,y) \rightarrow p^+(x,y)$ and the transitivity rule for p^+. Thus, we have $\mathcal{F}_\ell, \mathcal{R} \models \mathcal{F}_{\ell+1}$.

We have thus shown that for every $0 \leq \ell < m$, $\mathcal{F}_\ell, \mathcal{R} \models \mathcal{F}_{\ell+1}$. Since $\mathcal{F}' = \mathcal{F}_0$, by chaining these implications together, we obtain $\mathcal{F}', \mathcal{R} \models \mathcal{F}_m$. Using the same reasoning as above, we can infer $\mathcal{F}_m \models Q'$ from $\mathcal{F}_m^{++} \models (Q')^{++}$. Then, by combining these statements, we obtain $\mathcal{F}', \mathcal{R} \models Q'$. *(end proof of claim)*

Now let $\mathbb{Q}$ be the set of queries computed in Step 5 by performing all possible external direct rewritings w.r.t. $\mathbb{P}$ and rules from $\mathcal{R}_L^+$, starting from Q^+, and let $Q_{\mathbb{Q}}$ be the set of CQs associated with $\mathbb{Q}$ (defined as in Step 6). We start by proving the following claim, which relates external direct rewriting steps to sequences of classical direct rewritings.

Claim 4.2 *Let $\mathcal{Q}_{i+1}$ be a direct rewriting of $\mathcal{Q}_i$ w.r.t. $\mathbb{P}$. Then every full instance of $(\mathcal{Q}_{i+1}, \mathbb{P})$ is obtained from a sequence of (classical) direct rewritings of some full instance of $(\mathcal{Q}_i, \mathbb{P})$.*

Proof of claim. Let $(\mathcal{Q}_{i+1}, \mathbb{P})$ be obtained from an external rewriting of $(\mathcal{Q}_i, \mathbb{P})$ with rule $R = B \rightarrow H$. This means that there is a minimally unifiable instance $\mathcal{Q}^e$ and a unifier $\mu = (X, H, P_u)$ of $\mathcal{Q}^e$ with H (with associated substitution σ) such that $\mathcal{Q}_{i+1} = \sigma(\mathcal{Q}^e \setminus X) \cup \sigma(B)$. Let us consider a partial instance $\mathcal{Q}_{i+1}^P$ of $(\mathcal{Q}_{i+1}, \mathbb{P})$ that fully instantiates $\sigma(\mathcal{Q}^e \setminus X)$ but does not instantiate $\sigma(B)$ (we say that it is a $\sigma(B)$-excluding instance). Note that $\mathcal{Q}_{i+1}^P$ can be built equivalently by choosing a full instance Q^e of $(\mathcal{Q}^e, \mathbb{P})$, removing the atoms of X, then by applying the substitution σ and adding $\sigma(B)$. We can see that the classical direct rewriting of Q^e according to μ produces $\mathcal{Q}_{i+1}^P$. Moreover, since every full instance of $(\mathcal{Q}^e, \mathbb{P})$ is a full instance of $(\mathcal{Q}_i, \mathbb{P})$, we know that Q^e is an instance of $(\mathcal{Q}_i, \mathbb{P})$.

Now consider any full instance Q_{i+1} of $(\mathcal{Q}_{i+1}, \mathbb{P})$. Note that it is a full instance of some $\sigma(B)$-excluding instance $(\mathcal{Q}_{i+1}^P, \mathbb{P})$. There are two cases to consider:

- If $\sigma(B)$ is an atom, then $Q_{i+1} = \mathcal{Q}_{i+1}^P$ and thus Q_{i+1} is obtained from a classical direct rewriting of an instance of $(\mathcal{Q}_i, \mathbb{P})$.

- Otherwise, if $\sigma(B)$ is a repeatable pattern, then Q_{i+1} is obtained from $(\mathcal{Q}_{i+1}^P, \mathbb{P})$ by expanding $\sigma(B)$ into a sequence of k standard patterns, and expanding each of them into some atom a_ℓ. Let $B_k = \{a_\ell \mid 1 \leq \ell \leq k\}$. Then, $\sigma(B)$ is generated in forward chaining from B_k with a sequence of applications of rules: k applications of transitivity rules, and k applications of the rules encoded in $\mathbb{P}$, each one stemming from a finite sequence of applications of rules of $\mathcal{R}$ (see Claim 4.1). Thus from the completeness of classical rewriting, B_k can be obtained from a sequence of classical direct rewritings from $\sigma(B)$, and thus Q_{i+1} is obtained from a sequence of classical direct rewritings of an instance of $(\mathcal{Q}_i, \mathbb{P})$.

(end proof of claim)

The following claim shows the soundness of the external rewriting in Step 5 and completes the proof of the lemma.

Claim 4.3 *If* $\mathcal{F}, \Pi_{\mathbb{P}} \models Q_{\mathbb{Q}}$, *then* $\mathcal{F}, \mathcal{R} \models Q$.

Proof of claim. Suppose that $\mathcal{F}, \Pi_{\mathbb{P}} \models Q_{\mathcal{Q}}$ with $\mathcal{Q} \in \mathbb{Q}$. We know that the PCQ $\mathcal{Q}$ is obtained from a finite sequence $\mathcal{Q}_0 = Q^+, \mathcal{Q}_1, \ldots, \mathcal{Q}_k = \mathcal{Q}$ of PCQs such that for all $0 \leq j < k$, $(\mathcal{Q}_{j+1}, \mathbb{P})$ is a direct external rewriting of $(\mathcal{Q}_j, \mathbb{P})$. We will show by induction on j that $\mathcal{F}, \Pi_{\mathbb{P}} \models Q_{\mathcal{Q}_j}$ implies $\mathcal{F}, \mathcal{R} \models Q$ for every $0 \leq j \leq k$.

The base case ($j = 0$) is a direct consequence of Claim 4.1. For the induction step, we assume the property is true at rank i, and we show that it is true at rank $i + 1$.

Suppose that $\mathcal{F}, \Pi_{\mathbb{P}} \models Q_{\mathcal{Q}_{i+1}}$. From Proposition 4.2, it follows that there is a full instance Q_{i+1} of $(\mathcal{Q}_{i+1}, \mathbb{P})$ such that $\mathcal{F} \models Q_{i+1}$. By Claim 4.2, there is a full instance Q_i of $(\mathcal{Q}_i, \mathbb{P})$ such that Q_{i+1} is obtained from a sequence of classical rewritings from Q_i. Thus (from the correctness of the classical rewriting), there is a set of facts $\mathcal{F}'$ such that $\mathcal{F}, \mathcal{R} \models \mathcal{F}'$ and $\mathcal{F}' \models Q_i$. Applying Proposition 4.2, we obtain $\mathcal{F}', \Pi_{\mathbb{P}} \models Q_{\mathcal{Q}_i}$. Now from our induction hypothesis, it follows that $\mathcal{F}', \mathcal{R} \models Q$, hence $\mathcal{F}, \mathcal{R} \models Q$. *(end proof of claim)* □

We can now immediately conclude with the proof of Theorem 4.6:
Proof: Follows from Lemma 4.4 and 4.5. □

Regarding termination, we observe that Step 3 (internal rewriting) must halt since every direct rewriting step adds a new atom (using a predicate from $\mathcal{R}_L^+$) to a pattern definition, and there are finitely many such atoms (up to isomorphism).

By contrast, Step 5 (external rewriting) may not halt, as the rewritings may grow unboundedly in size (where the size is the number of atoms and patterns). Thus, to ensure termination, we will modify Step 5 to exclude direct rewritings that increase the rewriting size. We first recall the four possible replacements of a repeatable pattern $P^+[t_1, t_2]$ when computing the minimally-unifiable instances (see Definition 4.12):

(i) $P^+[t_1, x_1] \wedge X[x_1, x_2] \wedge P^+[x_2, t_2]$

(ii) $P^+[t_1, x_1] \wedge X[x_1, t_2]$

(iii) $X[t_1, x_2] \wedge P^+[x_2, t_2]$

(iv) $X[t_1, t_2]$

We identify the following "problematic" minimally-unifiable instances, problematic in the sense that they may lead to direct rewritings larger than the original query:

1. Q' is composed of atoms expanded from a single repeatable pattern $P^+[t_1, t_2]$, $\sigma_{\mu'}(t_1) = \sigma_{\mu'}(t_2)$, and $P^+[t_1, t_2]$ is replaced as in case (i), (ii) or (iii).

2. Q' is completely obtained from the expansion of one or more repeatable patterns (i.e., there is no standard atom involved in the unifier), a term t of Q is unified with an existential variable of the head of the rule, t appears only in repeatable patterns of the form $P_i^+[t_i, t]$ (resp. $P_i^+[t, t_i]$), and all these repeatable patterns are rewritten as in case *(ii)* $P_i^+[t_i, t_i'] \wedge X[t_i', t]$ (resp. as in case *(iii)* $X[t, t_i'] \wedge P_i^+[t_i', t_i]$).

Note that in Point 2, we need not to exclude rewritings in which Q' contains at least one standard atom, since such atoms will be erased by the rewriting process.

We call a direct rewriting *excluded* if it is based on such a minimally-unifiable instance; otherwise, it is *non-excluded*.

Example 4.10
The rewriting $\mathcal{Q}_1'$ from Example 4.9 is excluded because it is obtained from the minimally-unifiable instance Q_1 in which the repeatable pattern $P_1^+[a, z]$ is expanded as in case (ii) and $P_2^+[z, b]$ as in case (iii), and z is unified with the existential variable z'.

In the following we first show that using only non-excluded direct rewritings ensures termination (Proposition 4.6). Then, we show that ignoring direct rewritings based on a minimally-unifiable instance as in Point 1 never compromises the completeness of our algorithm. Indeed the query obtained is always more specific than the rewritten query (Lemma 4.6). Finally, we show that under a safety condition, ignoring the direct rewritings based on a minimally-unifiable instance as in Point 2 also preserves completeness (Lemma 4.7).

Proposition 4.6

Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ and $R \in \mathcal{R}_L^+$. If $\mathcal{Q}'$ is a non-excluded direct rewriting of $\mathcal{Q}$ with R, then $|\mathcal{Q}'| \leq |\mathcal{Q}|$.

Proof: Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ, $R = (B, H) \in R_L^+$, Q be a non-excluded minimally-unifiable instance of $(\mathcal{Q}, \mathbb{P})$, $\mu = (Q', H, P_\mu)$ be an external unifier of Q with R, and σ_μ be a substitution induced by P_u.

Note that all repeatable patterns $P^+[t_1, t_2]$ are either untouched by the unifier, or replaced by the sequence X needed by the unifier (i.e., $X \subseteq Q'$), plus potentially a single repeatable pattern $P^+[t_1, x_1]$ (or $P^+[x_k, t_2]$). Indeed, the only situation that would lead us to introduce more than one repeatable pattern (i.e., as in External Rewriting case (i)) is when either t_1 or t_2 is unified with an existential variable.

However, if t_1 (or t_2) is unified with an existential variable, because of the piece condition on unifiers, no unifier of $P^+[t_1, x_1] \wedge X \wedge P^+[x_k, t_2]$ can be found.

Since $|B| = 1$, we have to show that all atoms that were introduced when replacing a repeatable pattern are erased by the direct rewriting of Q w.r.t. μ.

If Q' consists of at least one atom that is not expanded from a pattern, the direct rewriting of Q w.r.t. μ erases this atom as well as the X sequence.

Next assume Q' consists only of atoms expanded from repeatable patterns. If $Q' = \{P^+[t_1, t_2]\}$ and neither t_1 nor t_2 is unified with an existential variable, then $\sigma_\mu(t_1) = \sigma_\mu(t_2)$, so the only non-excluded minimally-unifiable instance of $\mathcal{Q}$ w.r.t. μ replaces $P^+[t_1, t_2]$ only by the sequence X needed by the unifier (see the first condition on non-excluded minimally-unifiable instances). Thus, the direct rewriting erases $P^+[t_1, t_2]$.

Otherwise, we know that at least one $P^+[t_1, t_2]$ from $\mathcal{Q}$ is replaced by the sequence X involved in the unifier (see the second condition on non-excluded minimally-unifiable instances), thus there is at least one $P^+[t_1, t_2]$ erased by the direct rewriting. □

Let us consider the "modified query rewriting algorithm" that is obtained by only performing non-excluded direct rewritings in Step 5 (External Rewriting). This modification ensures termination but may comprise completeness. However, we can show that the modified algorithm is complete in the following key cases: when the conjunctive query is atomic, when there is no specialisation of a transitive predicate, or when all predicates have arity at most two. These cases rely on the following different observations. In the first case, completeness cannot be compromise since there is a single problematic case detailed in Lemma 4.7, and it cannot happen for atomic query since it involves two different repeatable patterns. In the second case, there is no rewriting of transitive atoms. Finally in the third case, the external terms t_1 and t_2 from each repeatable pattern are always unified together, ensuring that for each excluded rewriting, the obtained PCQ is more specific than the rewritten query. By further analyzing the latter case, we can formulate a safety condition, defined next, that guarantees completeness for a much wider class of rule sets.

We begin by defining a specialisation relationship between predicates. A predicate q is a *direct specialisation* of a binary predicate p on positions $\{\vec{i}, \vec{j}\}$ ($\vec{i} \neq \emptyset, \vec{j} \neq \emptyset$) if there is a rule of the form $q(\vec{u}) \to p(x, y)$ such that $\vec{i}$ (resp. $\vec{j}$) contains those positions of $\vec{u}$ that contain the term x (resp. y). It is a *specialisation* of p on positions $\{\vec{i}, \vec{j}\}$ if (a) it is a direct specialisation of p on positions $\{\vec{i}, \vec{j}\}$, or (b) there is a rule of the form $q(\vec{u}) \to r(\vec{v})$ such that $r(\vec{v})$ is a specialisation of p on positions $\{\vec{k}, \vec{l}\}$ and the terms occurring in positions $\{\vec{k}, \vec{l}\}$ of $\vec{v}$ occur in positions $\{\vec{i}, \vec{j}\}$ of $\vec{u}$ with $\vec{i} \neq \emptyset$ and $\vec{j} \neq \emptyset$. We say that q is a *pseudo-transitive predicate* if it is a specialisation of at least one transitive predicate.

Definition 4.13 (Safe Rule Set)
We call a linear+trans *rule set* safe *if it satisfies the following* safety condition*: for*

every pseudo-transitive predicate q, there exists a pair of positions $\{i, j\}$ with $i \neq j$ such that for all transitive predicates p of which q is a specialisation on positions $\{\vec{i}, \vec{j}\}$, either $i \in \vec{i}$ and $j \in \vec{j}$, or $i \in \vec{j}$ and $j \in \vec{i}$.

Note that if we consider binary predicates, the safety condition is always fulfilled. Then, specialisations correspond exactly to the subroles extended to inverses considered in Description Logics.

Example 4.11
Let $R_1 = s_1(x, x, y) \rightarrow p_1(x, y)$, $R_2 = s_2(x, y, z) \rightarrow p_2(x, y)$, $R_3 = s_1(x, y, z) \rightarrow s_2(z, x, y)$, and p_1 and p_2 be two transitive predicates.

The following specialisations have to be considered: s_1 is a direct specialisation of p_1 on positions $\{\{1, 2\}, \{3\}\}$, s_2 is a direct specialisation of p_2 on positions $\{\{1\}, \{2\}\}$, s_1 is a specialisation of p_2 on positions $\{\{3\}, \{1\}\}$. We then have two pseudo-transitive predicates: s_1 and s_2. By choosing the pair $\{1, 3\}$ for s_1 and $\{1, 2\}$ for s_2, we observe that $\{R_1, R_2, R_3\}$ satisfies the safety condition.

If we replace R_3 by $R_4 = s_1(x, y, z) \rightarrow s_2(x, y, z)$, s_1 is a specialisation of p_2 on positions $\{\{1\}, \{2\}\}$, and $\{R_1, R_2, R_4\}$ is not safe.

Theorem 4.7

The modified query rewriting algorithm halts. Moreover, Theorem 4.6 (soundness and completeness) holds for the modified algorithm if either the input conjunctive query is atomic, or the input rule set is safe.

We show this theorem with the help of the following two lemmas that ensure that the excluded rewritings are not necessary in the case of safe rule sets, or atomic conjunctive queries.

Lemma 4.6

Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ, $R \in R_L^+$, Q be an instance of interest of $(\mathcal{Q}, \mathbb{P})$ and $\mu = (Q', H, P_u)$ be an external unifier of Q with R such that two external terms w.r.t. μ from a given pattern $P^+[t_1, t_2]$ are unified together and with no existential variable.
Every minimally-unifiable instance $(\mathcal{Q}, \mathbb{P})$ w.r.t. μ that replaces $P^+[t_1, t_2]$ as in the External Rewriting cases (i), (ii), or (iii) leads to a direct rewriting $(\mathcal{Q}'_i, \mathbb{P})$ that is more specific than $(\mathcal{Q}, \mathbb{P})$.
Furthermore, for any classical direct rewriting $\mathcal{Q}''_i$ of $\mathcal{Q}'_i$ with R, either $(\mathcal{Q}, \mathbb{P}) \geq (\mathcal{Q}''_i, \mathbb{P})$ or there is a classical direct rewriting $\mathcal{Q}'$ of the minimally-unifiable instance of $\mathcal{Q}$ that replaces $P^+[t_1, t_2]$ as in case (iv) with R such that $(\mathcal{Q}', \mathbb{P}) \geq (\mathcal{Q}''_i, \mathbb{P})$.

Proof: Without loss of generality, let us write $\mathcal{Q} = q[t_1, t_2] \wedge P^+[t_1, t_2]$ where $q[t_1, t_2]$ denotes a set of atoms where t_1 and t_2 may occur. We denote by x_s and x_e $(s < e)$ the external terms of $P^+[t_1, t_2]$ w.r.t. μ, and by $A[x_s, x_e]$ the sequence of atoms expanded from $P^+[t_1, t_2]$ involved in the unifier. Since we assume that no existential variable is unified with variables x_s and x_e, no atom from $q[t_1, t_2]$ can be part of the unifier. Consider the following minimally-unifiable instances:

1. $\mathcal{Q}_1 = q[t_1, t_2] \wedge P^+[t_1, x_s] \wedge A[x_s, x_e] \wedge P^+[x_e, t_2]$
2. $\mathcal{Q}_2 = q[t_1, t_2] \wedge A[t_1 = x_s, x_e] \wedge P^+[x_e, t_2]$
3. $\mathcal{Q}_3 = q[t_1, t_2] \wedge P^+[t_1, x_s] \wedge A[x_s, x_e = t_2]$

By unifying x_s and x_e together, we obtain the following instances:

1. $q[t_1, t_2] \wedge P^+[t_1, x_s] \wedge A[x_s, x_s] \wedge P^+[x_s, t_2]$
2. $q[t_1, t_2] \wedge A[t_1, t_1] \wedge P^+[t_1, t_2]$
3. $q[t_1, t_2] \wedge P^+[t_1, t_2] \wedge A[t_2, t_2]$

Let $\mathcal{Q}'_i$ be the direct rewriting of $\mathcal{Q}_i$ w.r.t. μ with R. It is easy to see that $\mathcal{Q} \subseteq \mathcal{Q}'_2$ and $\mathcal{Q} \subseteq \mathcal{Q}'_3$, thus, $(\mathcal{Q}'_2, \mathbb{P})$ and $(\mathcal{Q}'_3, \mathbb{P})$ are more specific than $(\mathcal{Q}, \mathbb{P})$.

Let Q_1 be a full instance of $(\mathcal{Q}'_1, \mathbb{P})$. We construct a full instance Q of $(\mathcal{Q}, \mathbb{P})$ as follows. First note that $q[t_1, t_2]$ is common to both $\mathcal{Q}'_1$ and $\mathcal{Q}$, so we expand all patterns in $q[t_1, t_2]$ exactly as in Q_1. Now let k_1 (resp. k_2) be the number of children of $P^+[t_1, x_s]$ (resp. $P^+[x_s, t_2]$) in the instantiation of Q_1, and expand $P^+[t_1, t_2]$ in Q into $k = k_1 + k_2$ children: $P[t_1 = x_0, x_1], \ldots, P[x_{k-1}, x_k = t_2]$. Expand each $P[x_i, x_{i+1}]$ with $i < k_1$ as is expanded the i^{th} child of $P^+[t_1, x_s]$ in Q_1; and each $P[x_i, x_{i+1}]$ with $k_1 \leq i < k$ as is expanded the $(i - k_1 + 1)^{th}$ child of $P^+[x_s, t_2]$ in Q_1. By construction, there is a homomorphism from Q to Q_1. We have thus shown that $(\mathcal{Q}, \mathbb{P}) \geq (\mathcal{Q}'_1, \mathbb{P})$.

Furthermore, let $\mathcal{Q}''_i$ be a classical direct rewriting of $\mathcal{Q}'_i$ with a rule $R' = B' \to H'$ w.r.t. unifier $\mu' = (Q', H', P'_u)$, where $1 \leq i \leq 3$. If at least one atom involved in μ' occurs in $\mathcal{Q}'_i \setminus \sigma(B)$ (where σ is the substitution associated with μ), then, let $\mu'' = \{Q'', H', P''_u\}$ where $Q'' = Q' \setminus \sigma(B)$ and P''_u is the restriction of P'_u to terms occurring in $Q'' \cup H'$. Since $Q'' \neq \emptyset$ and all terms from $\sigma(B)$ cannot connect two different terms from $q[t_1, t_2]$ (indeed, the only term shared between $\sigma(B)$ and $q[t_1, t_2]$ is either t_1 or t_2), $\sigma(B)$ can be seen as a loop on t_1 (or t_2), therefore we can remove $\sigma(B)$ while preserving the unifier, i.e., μ'' is a unifier of $\mathcal{Q}$ with R'. Moreover, since P''_u and Q'' are only restrictions of P'_u and Q' respectively, it holds that $\mu'' \geq \mu'$. Then, we denote by $\mathcal{Q}''$ the direct rewriting of $\mathcal{Q}$ with R' w.r.t. μ'' and obtain $\mathcal{Q}'' \geq \mathcal{Q}'_i$. The other possibility is that all atoms involved in μ' occur in $\sigma(B)$, then, $\mathcal{Q}''_2$ (resp. $\mathcal{Q}''_3$) is more specific than $\mathcal{Q}$ since $\mathcal{Q} \subseteq \mathcal{Q}''_2$ (resp. $\mathcal{Q} \subseteq \mathcal{Q}''_3$). Moreover, for any instance Q''_1 of $\mathcal{Q}''_1$, one can easily build an instance Q' of $\mathcal{Q}$ in the same way as above, and see that $Q' \geq Q''_1$. Thus, we have $(\mathcal{Q}, \mathbb{P}) \geq (\mathcal{Q}''_i, \mathbb{P})$. □

Lemma 4.7

Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ, $R \in \mathcal{R}_L^+$, Q be an instance of interest of $(\mathcal{Q}, \mathbb{P})$ and $\mu = (Q', H, P_u)$ be an external unifier of Q with R such that one external term w.r.t. μ from a given pattern $P^+[t_1, t_2]$ is unified with an existential variable, and where all atoms in Q' are obtained from the expansion of a repeatable pattern.
If $\mathcal{Q}$ is atomic, or if $\mathcal{R}_L$ is a set of safe linear rules, then every minimally-unifiable instance of $(\mathcal{Q}, \mathbb{P})$ w.r.t. μ that replaces all $P_i^+[t_1, t_2]$ as in the External Rewriting cases (ii) or (iii) leads to a direct rewriting $(\mathcal{Q}'_i, \mathbb{P})$ that is more specific than $(\mathcal{Q}, \mathbb{P})$.
Furthermore, for any direct rewriting $\mathcal{Q}''_i$ of $\mathcal{Q}'_i$ with R, either $\mathcal{Q} \geq \mathcal{Q}''_i$ or there is a direct rewriting $\mathcal{Q}'$ of a minimally-unifiable instance of $\mathcal{Q}$ w.r.t. μ that replaces at least one repeatable pattern as in case (iv) and is such that $\mathcal{Q}' \geq \mathcal{Q}''_i$.

Proof: Let $(\mathcal{Q}, \mathbb{P})$, R, Q and μ be as in the lemma statement, and let $P_1^+[t_1^1, t_2^1], \ldots, P_k^+[t_1^k, t_2^k]$ be the repeatable patterns that are relevant for μ. For each $1 \leq i \leq k$, we denote by $P_i[t_1^i = x_0^i, x_1^i], \ldots, P_i[x_{k-1}^i, x_{k_i}^i = t_2^i]$ the sequence of standard patterns expanded from $P_i^+[t_1^i, t_2^i]$, and we let $x_{s_i}^i$ and $x_{e_i}^i$ $(s_i < e_i)$ be the external terms of $P_i^+[t_1^i, t_2^i]$ w.r.t. μ. We assume without loss of generality that it is $x_{e_i}^i$ that is unified with an existential variable, and let $A_i[x_{s_i}^i, x_{e_i}^i = t_2^i]$ denote the atoms expanded from $P_i[x_j^i, x_{j+1}^i]$ with $s_i \leq j < e_i$.

Since the unifier μ is single-piece, and no pattern is expanded as in case (iv) all repeatable patterns relevant for μ have to share some variable. For simplicity, we assume that they all share their second term, i.e. $t_2^i = t_2^j$ for all $1 \leq i, j \leq k$. (The argument is entirely similar, just more notationally involved, if this assumption is not made.) Let us use t_2 for this shared term. Then we can write $\mathcal{Q}$ as follows:

$$\mathcal{Q} = q[t_1^1, \ldots, t_1^k] \wedge \bigwedge_{1 \leq i \leq k} P_i^+[t_1^i, t_2]$$

Note that t_2 cannot occur in q.

Because we have chosen the second term to be shared in all repeatable patterns, we only need to consider the minimally-unifiable instance $\mathcal{Q}_M$ of $(\mathcal{Q}, \mathbb{P})$ w.r.t. μ that replaces each $P_i^+[t_1^i, t_2]$ by $P_i^+[t_1^i, x_s^i], A_i[x_s^i, x_e^i = t_2]$, i.e. External Rewriting case (ii). Thus, we have

$$\mathcal{Q}_M = q[t_1^1, \ldots, t_1^k] \wedge \bigwedge_{1 \leq i \leq k} (P_i^+[t_1^i, x_s^i] \wedge A_i[x_s^i, t_2]).$$

Let σ be the substitution associated with μ. From the safety condition, we know that there is a pair of positions $\{p_1, p_2\}$ for the predicate p of H, such that for all

atoms $p(\vec{t})$ occurring in a pattern definition the terms #1 and #2 occurs in positions $\{p_1, p_2\}$. We further note that the external terms in the concerned patterns are t_2 (which unifies with an existential variable in H) and the terms x_s^i (which unify with a non-existential variable), and each of these external terms must be obtained by instantiating term #1 or #2. Since the $A_i[x_s^i, t_2]$ are unified together, and share the same predicate p, it follows that all of the x_s^i must occur in the same position (either p_1 or p_2) of p; t_2 occurs in the other position among p_1 and p_2. We therefore obtain;

$$\sigma(x_s^1) = \sigma(x_s^2) = \cdots = \sigma(x_s^k) = x',$$

where x' is the term in B that unifies with all of the x_s^i. (Note that if $\mathcal{Q}$ is an atomic query, there is a single A_i, so the previous statement obviously holds, even without the safety condition.) Thus, $\mathcal{Q}_M$ becomes:

$$q[t_1^1, \ldots, t_1^k] \wedge \bigwedge_{1 \leq i \leq k} (P_i^+[t_1^i, x'] \wedge A_i[x', t_2]).$$

There is an isomorphism from $\mathcal{Q}$ to $\mathcal{Q}_M \setminus \{A_i \mid 1 \leq i \leq k\}$ that maps t_2 to x'. We then observe that $\{A_i \mid 1 \leq i \leq k\}$ is exactly the set of atoms that will be erased in the direct rewriting $\mathcal{Q}'_M = \mathcal{Q}_M \setminus \{A_i \mid 1 \leq i \leq k\} \cup \sigma(B)$, where σ is a substitution associated with μ. Therefore, $\mathcal{Q}$ is isomorphic to $\mathcal{Q}'_M \setminus \sigma(B)$, hence $(\mathcal{Q}, \mathbb{P}) \geq (\mathcal{Q}'_M, \mathbb{P})$. One can see that the same reasoning as in the previous proof can be applied here to show that any further direct rewriting $\mathcal{Q}''_M$ of $\mathcal{Q}'_M$ will lead to more specific queries. □

We are now ready to prove Theorem 4.7.

Proof: Termination is a direct consequence of Proposition 4.6, indeed, since during the rewriting process we only keep PCQs that are not isomorphic to others, and Proposition 4.6 ensures that the size of each rewriting is bounded by the size of the initial query, there will be only finitely many such rewritings.

Regarding correctness, we know from Lemma 4.4 that if we do not exclude any rewriting the algorithm is sound and complete, and Lemma 4.6 and 4.7 show that for any rewriting $\mathcal{Q}$ that we exclude, there is another rewriting $\mathcal{Q}'$ obtainable using only non-excluded direct rewritings that is more general than $\mathcal{Q}$. Therefore, the modified algorithm (in case of an atomic CQ, or a safe rule set) is complete. Furthermore, excluding rewritings cannot compromise the soundness of the rewriting mechanism. □

4.2.5 Complexity

A careful analysis of our query rewriting algorithm allows us to compute the worst-case complexity of atomic conjunctive query entailment over *linear+trans* knowledge bases, and general conjunctive query entailment over safe *linear+trans* knowledge

bases (with still an open question as explained later). We consider two different complexity measures: *combined complexity* (that considers the size of the whole input, i.e., the knowledge base and the query), and *data complexity* (that assumes that only the set of facts is part of the input). The latter is often considered more relevant since the set of facts is typically significantly larger than the rest of the input.

Let us first recall that the complexity of entailment over linear knowledge bases is *PSpace*-complete in combined complexity (even with atomic CQs) [CGL$^+$10b] and in AC_0 in data complexity (since they are FUS); while regarding entailment over transitivity knowledge bases (i.e., where all the rules are transitivity rules), it is complete for non-deterministic logarithmic space (NL) in data complexity (see, e.g., [Pap94] for the problem of reachability in directed graph), and NP-complete in combined complexity (see Table 4.1).

We show that the complexity of the combination is the best that we could hope with regards to data complexity. Indeed, we show completeness for NL, hence the same complexity as in the presence of transitivity rules alone. The situation is different for combined complexity, where the complexity for atomic CQ entailment increases from *PSpace* for linear rules alone to *ExpTime* for linear rules with transitivity.

Theorem 4.8 (Data Complexity)

Both atomic conjunctive query entailment over *linear+trans* knowledge bases, and conjunctive query entailment over safe *linear+trans* knowledge bases are NL-complete in data complexity.

Proof: Consider a CQ Q, a *linear+trans* rule set $\mathcal{R}$, and a set of facts $\mathcal{F}$. Suppose that either Q is atomic or $\mathcal{R}$ satisfies the safety condition. Using Theorem 4.7, we can compute a finite set $\Pi_{\mathbb{P}}$ of Datalog rules and a finite set $Q_{\mathbb{Q}}$ of CQs with the property that $(\mathcal{F}, \mathcal{R}) \models Q$ iff $(\mathcal{F}, \Pi_{\mathbb{P}}) \models Q'$ for some $Q' \in Q_{\mathbb{Q}}$. As $\Pi_{\mathbb{P}}$ and $Q_{\mathbb{Q}}$ do not depend on the set of facts $\mathcal{F}$, they can be computed and stored using constant space w.r.t. $|\mathcal{F}|$.

To test whether $(\mathcal{F}, \Pi_{\mathbb{P}}) \models Q'$ for some $Q' \in Q_{\mathbb{Q}}$, we proceed as follows. For each rewriting $Q' \in Q_{\mathbb{Q}}$, we can consider every possible mapping π from the variables of Q' to the terms of $\mathcal{F}$. We then check whether the facts in $\pi(Q')$ are entailed from $(\mathcal{F}, \Pi_{\mathbb{P}})$. For every atom $\alpha \in Q'$ over one of the original predicates, we can directly check if $\pi(\alpha) \in \mathcal{F}$, since the rules in $\Pi_{\mathbb{P}}$ can only be used to derive facts over the new predicates p^+. For every atom $p^+[t_1, t_2] \in Q'$ where p^+ is a new predicate, we need to check whether $(\mathcal{F}, \Pi_{\mathbb{P}}) \models p^+(\pi(t_1), \pi(t_2))$. Because of the shape of the rules in $\Pi_{\mathbb{P}}$, the latter holds just in the case that there is a path of constants $c_1, \ldots, c_n$ with $c_1 = \pi(t_1)$ and $c_n = \pi(t_2)$ such that for every $1 \leq i < n$, there is a rule $\rho_i = B_i \to p^+(\#1, \#2)$ and substitution σ_i of the variables in B_i by constants in $\mathcal{F}$

such that $\sigma_i(\#1) = c_i$, $\sigma_i(\#2) = c_{i+1}$, and $\sigma_i(B_i) \in \mathcal{F}$. To check for the existence of such a path, we guess the constants c_i in the path one at a time, together with the witnessing rule ρ_i and substitution σ_i, using a counter to ensure that the number of guessed constants does not exceed the number of constants in $\mathcal{F}$. Note that we need only logarithmically many bits for the counter, so the entire procedure runs in non-deterministic logarithmic space.

Hardness for NL can be shown by an easy reduction from the NL-complete directed reachability problem (using only transitivity rules) to atomic conjunctive query entailment over safe (linear+)trans knowledge bases. We recall the definition of Reachability from [Pap94]: given a (directed) graph $G = (V, E)$ and two vertices $s, t \in V$, is there a path from s to t in G?

The reduction is as follows. Regarding the vocabulary, we consider for each vertex $u \in V$ the constant c_u; and a single binary predicate e, that is used to first represent initial edges, and then paths. Now, for each (directed) edge $(u, v) \in E$, we add an atom $e(c_u, c_v)$ to $\mathcal{F}$. Next, $\mathcal{R}$ contains the single transitivity rule $e(x, y) \wedge e(y, z) \rightarrow e(x, z)$. Intuitively, this rule "propagates" the relation e on all accessible vertices, i.e., if this rule were to be applied until fixpoint, there would be $e(c_u, c_v)$ in the saturated set of facts if and only if the vertex v is reachable from the vertex u. Therefore, the query $e(c_s, c_t)$ is entailed by $(\mathcal{F}, \mathcal{R})$ if and only if there is a path from s to t in G. Moreover $\mathcal{R}$ is trivially safe. □

Theorem 4.9 (Combined Complexity Upper Bound)

Both (i) atomic conjunctive query entailment over *linear+trans* knowledge bases, and (ii) conjunctive query entailment over safe *linear+trans* knowledge bases are in *ExpTime* in combined complexity.

Proof: Consider a CQ Q, a *linear+trans* rule set $\mathcal{R} = \mathcal{R}_L \cup \mathcal{R}_T$, with $\mathcal{R}_L$ a set of linear rules and $\mathcal{R}_T$ a set of transitivity rules, and a set of facts $\mathcal{F}$. Suppose that either condition (i) or (ii) of the theorem statement holds. It follows from Theorem 4.7 that the modified query rewriting algorithm halts and returns a finite set $\Pi_{\mathbb{P}}$ of Datalog rules and a finite set $Q_{\mathbb{Q}}$ of CQs such that $(\mathcal{F}, \mathcal{R}) \models Q$ iff $(\mathcal{F}, \Pi_{\mathbb{P}}) \models Q'$ for some $Q' \in Q_{\mathbb{Q}}$.

To prove membership in *ExpTime*, we show that:

(i) $\Pi_{\mathbb{P}}$ is of exponential size and can be built in exponential time;

(ii) $Q_{\mathbb{Q}}$ is a set of exponential size, that can be built in exponential time, and any $Q' \in Q_{\mathbb{Q}}$ is of linear size in Q;

(iii) we can saturate $\mathcal{F}$ with $\Pi_{\mathbb{P}}$ into $\mathcal{F}^*$ in polynomial time in the size of $\Pi_{\mathbb{P}}$ and $\mathcal{F}$, and the resulting set of facts is of polynomial size in $\mathcal{F}$;

(iv) $Q_{\mathbb{Q}}$ can be evaluated over $\mathcal{F}^*$ in exponential time.

We denote by r the maximum arity of a predicate in $\mathcal{R}$, by p the number of predicates occurring in $\mathcal{R}$ and by t the number of transitive predicates.

Let us consider the construction of $\Pi_{\mathbb{P}}$. Since all rules generated in this step are linear rules and given a predicate s the number of non-isomorphic atoms using s is bounded by an exponential in r, for each transitive predicate there can be only exponentially many generated rules. Thus $|\Pi_{\mathbb{P}}| = O(t \times p \times r^r)$. For the first point, it remains to show that $\Pi_{\mathbb{P}}$ can be built in exponential time. Consider the following algorithm: for each pattern definition P, repeat until fixpoint: choose a rule $R = (B, H) \in \mathcal{R}_L$, compute all instances of interest of P w.r.t. R, and if there is an internal unifier, add the corresponding rewriting to P's definition. The repeatable pattern $P^+[t_1, t_2]$ can be expanded into at most $r + 2$ standard patterns (see Definition 4.10), and thus there are $r + 2$ possible sizes for the instances of interest. Then for each of these standard patterns, we can choose an atom from $P's$ definition that uses the predicate of H. Since there are at most r^r possible choices for instantiating a standard pattern and there are at most $r + 2$ standard patterns to expand, we obtain the following bound: there are $O((r + 2) \times (r^r)^{r+2}) = O(r^{r^2})$ different instances of interest for a given pattern definition and a given rule. Therefore each step of the algorithm can be processed in exponential time. Since there are only exponentially many different possible rewritings, the fixpoint is reached in at most exponential time. Hence, Point (i) runs in exponential time.

The argument for Point (ii) proceeds similarly. The only difference comes from the fact that since Q might not be atomic, we apply the rewriting step to conjunctive queries. However, from Proposition 4.6, we know that all rewritten queries have size bounded by the size of Q. Therefore, by using the same argument as for Point (i), we know that this step is exponential in both the maximum arity and in the size of the initial query Q.

Regarding Point (iii), a single breadth-first step with all non-transitive rules in $\Pi_{\mathbb{P}}$ followed by the computation of the transitive closure is enough to build $\mathcal{F}^*$. While there are exponentially many non-transitive rules, each can be applied in polynomial time (since the body of each rule is atomic). Since each rule only creates atoms with transitive predicates, the resulting set of facts is of size $|terms(\mathcal{F})|^2 \times p$. Now the transitive closure adds at most a quadratic number of atoms (for each transitive predicate), and can be computed in polynomial time in the size of $\mathcal{F}$. Therefore, $\Pi_{\mathbb{P}}$ can be built in exponential time in r and is of polynomial size in $|\mathcal{F}|$.

It remains to show that point (iv) can be done in exponential time. Observe that since each query $Q' \in Q_{\mathbb{Q}}$ is of size bounded by the initial query Q (Proposition 4.6), its evaluation can be computed in NP, thus in exponential time. Since there are only exponentially many queries in $Q_{\mathbb{Q}}$, this step is also done in exponential time.

Therefore, we can conclude that the entailment problem over *linear+trans* sets of rules with atomic query, and over safe *linear+trans* sets of rules is in $ExpTime$. □

Concerning *ExpTime*-hardness, we prove it for atomic CQ entailment over *linear+trans* knowledge bases, but the question is not solved yet for CQ entailment over safe *linear+trans* knowledge bases.

Theorem 4.10 (Combined Complexity Lower Bound)

Atomic conjunctive query entailment over *linear+trans* knowledge bases is *ExpTime*-hard in combined complexity.

Proof: To prove hardness, we can rely on a proof from [BT16]. In this paper, they prove that Regular-Path Query (RPQ) entailment over linear knowledge bases is *ExpTime*-hard. The problem is not a subproblem of ours, nor the contrary. However the proof uses only a particular RPQ of the form $p^+(t_1, t_2)$. This RPQ is entailed from $(\mathcal{F}, \mathcal{R}_L)$ if and only if the atomic CQ $p(t_1, t_2)$ is entailed from $(\mathcal{F}, \mathcal{R}_L \cup \{trans(p)\})$. Nevertheless, we recall below the main lines of the proof, while reformulating it in terms of our problem. Note that the linear rules have a non-atomic head to simplify the explanations, but can be decomposed into atomic-headed rules as detailed in Section 1.3.

The reduction is from the simulation of any Alternating Turing Machine (ATM) that runs in polynomial space. More specifically, the problem they consider is the following ExpTime-complete problem: given a PSpace ATM M, and a word x, does M accept x? Without loss of generality, they consider ATM where each non-final universal state has exactly two existential state successors, and each non-final existential state has exactly two universal state successors.

The proof uses a single transitive predicate that we call p. Given an ATM M with input x, we create a predicate of arity polynomial in x and M, that encodes the current configuration of the machine (its tape and the current state and head position). Furthermore, each atom encoding a configuration also uses a term as a "begin" and another as an "end" (respectively the first and last position of the predicate), these are used later by the transitivity rules. Linear rules are used to generate the transitions of the ATM. First, for each transition in the ATM, there is a linear rule that generates the two next configurations, and depending on the type of the current state different transitive atoms are generated as illustrated by Figure 4.5.

The initial configuration contains two special constants b and e as begin and end, and the set of facts contains only the atom encoding this configuration.

When the state of the current configuration s is existential, four atoms using predicate p are generated in the next step, the first two being used to link the begin of s to the begin of the two next configurations (since the ATM is non-deterministic by nature), and the last two atoms being used to link the end of the two next configurations to the end of s.

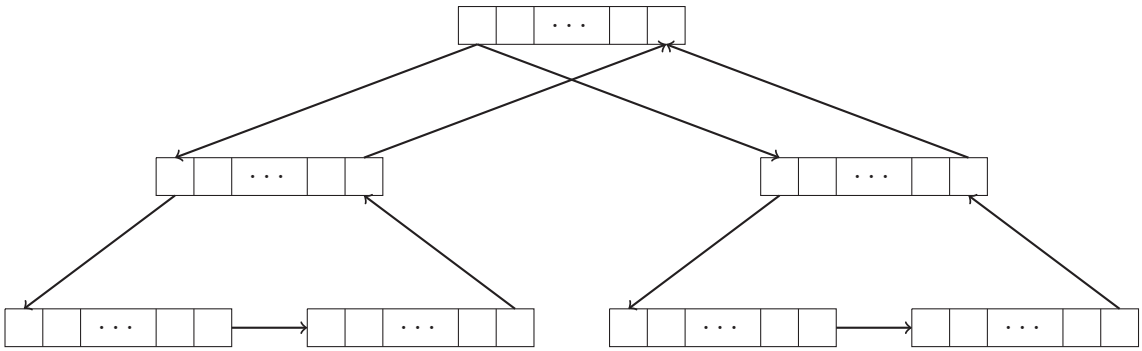

Figure 4.5: Reduction from ATM simulation to atomic CQ entailment over linear+*trans* knowledge bases. Edges stand for p-atoms and arrays stand for configuration atoms, with the first and last elements corresponding to the begin and end terms.

When the state of the current configuration s is universal, three atoms using p are generated, the first one links the begin of s to the begin of the first next configuration, the second one links the end of the first next configuration to the begin of the second next configuration, and finally the last one links the end of the last next configuration to the end of s.

Finally, when the state of the current configuration s is accepting, an atom using p linking the begin of s with the end of s is generated.

The idea is that linear rules simulate the run of the machine, and that transitivity rules connect the initial begin to the initial end if and only if M accepts x.

Then, the query just asks whether the begin of the initial configuration can be linked to the end of the initial configuration (i.e., $Q = p(b, e)$).

This reduction shows that atomic CQ entailment over *linear+trans* sets of rules is $ExpTime$-hard. □

From Theorems 4.9 and 4.10, we conclude that atomic CQ entailment over *linear+trans* knowledge bases is $ExpTime$-complete in combined complexity.

The previous reduction cannot be used to prove the ExpTime-hardness of CQ entailment over safe *linear+trans* knowledge bases. Indeed, the decomposition of the rules into atomic-headed rules produces unsafe linear rules. We point out that the rewriting of the set of *linear+trans* rules built in the previous reduction involves only internal rewritings. Since the safety condition was introduced to ensure the termination of the external rewriting step, one may conjecture that CQ entailment over safe *linear+trans* knowledge bases is ExpTime-hard. However it is not excluded that the safety condition reduces the complexity of the internal rewriting step. Note that the problem is at least PSpace-hard because it is already the case for linear rules alone (which are in this case, trivially safe).

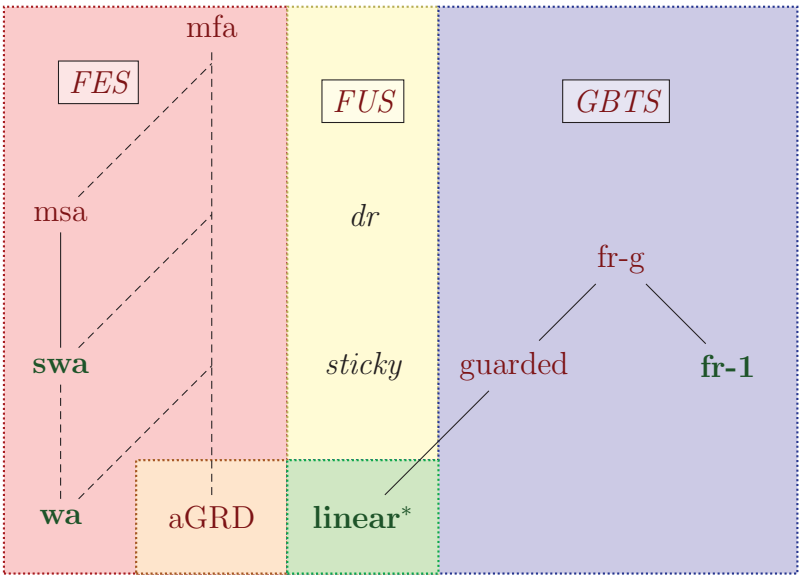

Figure 4.6: Decidability of Entailment over C+*trans* Knowledge Bases

Conclusion

In this chapter we have studied the decidability of entailment over knowledge bases built with some well-known decidable rule classes combined with transitivity rules. We first have recalled several known results from the litterature, in particular regarding specific cases of *BTS* rules, namely *guarded*, *frontier-guarded* and *frontier-1* rules. Then, we have shown that aGRD rules are not compatible with transitivity rules and thus neither *FES* nor *FUS* rules are. Furthermore, by also proving that MSA is not compatible with transitivity, we have drawn a complete picture of decidability results when combining known *FES* rule classes with transitivity. Then we have studied the case of *linear* rules, one of the simplest *FUS* rule classes, showing that they are compatible with transitivity up to a minor safety condition, and have provided complexity bounds in terms of data and combined complexities. These bounds are tight for data complexity (NL-complete). Concerning combined complexity, we have tight bounds in the case of atomic queries, namely ExpTime-completeness (in this case, the safety condition is not needed), and a membership result for conjunctive queries (up to the safety condition).

Figure 4.6 synthetises all those decidability results: rule classes that can be safely combined with transitivity are pictured in bold green, while those that cannot in red. In italic black font, we have classes of rules for which the question is still open. Furthermore, Table 4.1 describes the complexity of entailment for the positive cases.

Finally, some open questions still remain:

- What is the exact complexity of CQ entailment over safe *linear+trans* knowl-

Rule Class	**Combined**	**Data**
wa+trans	2ExpTime-c	PTime-c
swa+trans	2ExpTime-c	PTime-c
linear	PSpace-c	AC_0
fr-1	2ExpTime-c	PTime-c
trans	NP-c	NL-c
linear+trans	in ExpTime$^{(*)}$	NL-c
fr-1+trans	2ExpTime-c	PTime-c

($*$) ExpTime-complete for atomic CQ entailment

Table 4.1: Complexity of CQ Entailment over C*+trans* Knowledge Bases

edge bases? In particular, is it ExpTime-complete?

- Can we combine *linear* rules with transitivity without the safety condition while preserving decidability?
- What happens for *sticky* or *domain-restricted* rules, that also enjoy the *FUS* property, yet are incomparable with linear rules?

Beside these remaining decidability issues, the question of a practical algorithm for query answering with *linear+trans* knowledge bases deserves to be studied. Indeed, despite their simplicity, linear rules are an important class that allows to represent frequently used knowledge. Our algorithm for *linear+trans* is worst-case optimal with respect to complexity theory but would clearly not be efficient in practice.

Conclusion

In this thesis we considered the problem of querying data while exploiting general background knowledge, called an ontology. Ontological knowledge can be represented under different formalisms, and we have considered the existential rule framework.

In this setting, knowledge bases are composed of data, that can be abstracted as a conjunction of atoms, and ontological knowledge, represented thanks to existential rules, which are rules that have the specificity of having existentially quantified variables in their head. These variables allow to infer the existence of individuals not present in the initial data. This specificity has been considered as highly important in the context of Ontology-Mediated Query Answering (OMQA). Furthermore, these rules enjoy a great expressivity. Unsurprisingly, this expressivity has a cost: the main problem we consider, namely the conjunctive query entailment problem (CQ entailment), which asks whether a knowledge base entails a conjunctive query, is undecidable. However, many restrictions on the set of rules are known to restore decidability.

Summary of our Contributions

The contributions of this thesis can be split into two parts. On the one hand, we have recalled the various forward chaining algorithms, also known as chase variants, and proposed a uniform definition for all of them, pointing out their differences regarding finiteness. Then we have considered known rule classes for which some forward chaining algorithm halts. We observed that most of them rely on some acyclicity condition, and we have proposed a tool that has allowed us to unify their definitions. Then, thanks to this new tool, we have extended them, without increasing the (worst-case) complexity of the recognition. On the other hand, we have been interested in combining transitivity rules with known decidable classes of existential rules. Transitivity is acknowledged to be a fundamental property in ontological design, hence being able to take transitivity into account is important in this context. However, the previous results from the literature were not optimistic, as guarded rules (a well-known decidable class) were not compatible with transitivity. We first clarified the picture regarding FES rule classes (those for which the strongest forward chaining algorithm halts). Our undecidability result also shows that FUS classes (which ensure that there exists a sound and complete UCQ rewriting of

any query) are generally not compatible with transitivity. We have then focused on combining linear rules with transitivity. Linear rules are a subclass of guarded rules, but they are also FUS. We have shown that up to a minor safety condition, this combination is possible while preserving decidability. Finally, we have analysed the complexity of the CQ entailment over *linear+trans* knowledge bases, and have obtained tight results for the data complexity, as well as for the combined complexity in the case of atomic queries.

Perspectives

We have contributed to a better understanding of the landscape of decidable classes of rules. However, some interesting theoretical issues remain open.

With regard to the new acyclicity conditions, the complexity of their recognition has been characterised, but what about the complexity of CQ entailment with these rules? Do weaker acyclicity notions allow for a smaller complexity of reasoning? Another point is that the strongest acyclicity notion ensures the finiteness of the frontier-restricted (or skolem) chase, and no condition allows to ensure the finiteness of the core chase on knowledge bases for which the frontier-restricted chase may not halt. In other words, the acyclicity notions do not take the notion of redundancy into account.

Concerning transitivity, taking these rules into account appeared to be much more difficult than expected. Whether the safety condition on *linear+trans* set of rules is really needed is not clear. Moreover, the exact combined complexity of CQ entailment over safe *linear+trans* knowledge bases remains unknown. The compatibility of transitivity with other FUS classes is also still an open issue. Beyond transitivity, it could be nice to consider the composition of binary relations. Is it harder to process than transitivity? On the practical side, it is not clear how our algorithm for CQ entailment over *linear+trans* knowledge bases could be refined in order to lead to a practical algorithm. Notions like the instances of interest have been defined with worst-case complexity in mind, but clearly, the bounds on pattern expansions could be tighten. Furthermore, there is no need to generate all instances of interest each time we want to find a unifier with a PCQ, and we could try to build the needed instances of interest on the fly.

More generally, combining several paradigms seems promising, from both decidability and complexity points of view. The central idea is to use the rules on the one hand to rewrite the query and on the other hand to modify the set of facts, which may lead to new decidability results or to better complexity (typically, ensure that both processes can be performed in polynomial time). The so-called *combined approach* was introduced in the context of description logics, first for DL-lite knowledge bases [KLT$^+$09] (also [KLT$^+$10, LSTW13]), then for $\mathcal{EL}$ knowledge bases [KLT$^+$11, HLSW15]. The set of facts is "saturated" in a finite way (even when all universal models are infinite), which may lead to false positive answers to the query. Then the query is rewritten to avoid unwanted answers. Alternatively, false positive

answers to the initial query can be filtered out. The question of whether polynomial combination techniques are possible for various BTS existential rules is studied in [GMP14, GMP15]. While most results regarding guarded rules (and their generalisations) are negative, positive results are provided for linear rules and first-order query rewriting (instead of UCQ rewriting). Even if the proposed algorithm does not seem to be implementable, it is a first step that lets us hope for the best.

Finally, our work can also been extended with respect to the kinds of handled queries. Conjunctive queries (and unions of conjunctive queries) are usually considered in the OMQA setting. However, to make OMQA systems usable in practice, we have to consider more complex queries, like conjunctive queries integrating regular expressions (as e.g., in [BT16]), or more powerful relational queries.

Bibliography

[AB15] Antoine Amarilli and Michael Benedikt. Combining existential rules and description logics. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2691–2697, 2015.

[ABBV16] Antoine Amarilli, Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Query answering with transitive and linear-ordered data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 893–899, 2016.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Baa03] Franz Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 325–330, 2003.

[Bag04] Jean-François Baget. Improving the forward chaining algorithm for conceptual graphs rules. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 407–414, 2004.

[BBB+16a] Jean-François Baget, Salem Benferhat, Zied Bouraoui, Madalina Croitoru, Marie-Laure Mugnier, Odile Papini, Swan Rocher, and Karim Tabia. A general modifier-based framework for inconsistency-tolerant query answering. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 513–516, 2016.

[BBB+16b] Jean-François Baget, Salem Benferhat, Zied Bouraoui, Madalina Croitoru, Marie-Laure Mugnier, Odile Papini, Swan Rocher, and Karim Tabia. Inconsistency-tolerant query answering: Rationality

properties and computational complexity analysis. In *JELIA-16, Proceedings of the Nineteenth European Conference On Logics In Artificial Intelligence, 2016*, page to appear, 2016.

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 364–369, 2005.

[BBMR15] Jean-François Baget, Meghyn Bienvenu, Marie-Laure Mugnier, and Swan Rocher. Combining existential rules and transitivity: Next steps. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2720–2726, 2015.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BGL+15] Jean-François Baget, Alain Gutierrez, Michel Leclère, Marie-Laure Mugnier, Swan Rocher, and Clément Sipieter. Datalog+, ruleml and OWL 2: Formats and translations for existential rules. In *Proceedings of the RuleML 2015 Challenge, the Special Track on Rule-based Recommender Systems for the Web of Data, the Special Industry Track and the RuleML 2015 Doctoral Consortium hosted by the 9th International Web Rule Symposium (RuleML 2015), Berlin, Germany, August 2-5, 2015.*, 2015.

[BGMR14a] Jean-François Baget, Fabien Garreau, Marie-Laure Mugnier, and Swan Rocher. Extending acyclicity notions for existential rules. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 39–44, 2014.

[BGMR14b] Jean-François Baget, Fabien Garreau, Marie-Laure Mugnier, and Swan Rocher. Revisiting chase termination for existential rules and their extension to nonmonotonic negation. *CoRR*, abs/1405.1071, 2014.

[Bie16] Meghyn Bienvenu. Ontology-mediated query answering: Harnessing knowledge to get more from data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4058–4061, 2016.

[BLM10] Jean-François Baget, Michel Leclère, and Marie-Laure Mugnier. Walking the decidability line for rules with existential variables. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, 2010.

[BLM+15] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, Swan Rocher, and Clément Sipieter. Graal: A toolkit for query answering with existential rules. In *Rule Technologies: Foundations, Tools, and Applications - 9th International Symposium, RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings*, pages 328–344, 2015.

[BLMS09] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. Extending decidable cases for rules with existential variables. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 677–682, 2009.

[BLMS11] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

[BMRT11] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 712–717, 2011.

[BMRT14] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Worst-case optimal query answering for greedy sets of existential rules and their subclasses. *CoRR*, abs/1412.4485, 2014.

[BO15] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*, pages 218–307, 2015.

[Bod05] Manuel Bodirsky. The core of a countably categorical structure. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 110–120, 2005.

[BT16] Meghyn Bienvenu and Michaël Thomazo. On the complexity of evaluating regular path queries over linear existential rules. In *Web Rea-*

soning and Rule Systems - 10th International Conference, RR 2016, Aberdeen, UK, September 9-11, 2016, Proceedings, pages 1–17, 2016.

[BV81] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, pages 73–85, 1981.

[BV84] Catriel Beeri and Moshe Y. Vardi. Formal systems for tuple and equality generating dependencies. *SIAM J. Comput.*, 13(1):76–98, 1984.

[CCIL08] Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone. Computable functions in ASP: theory and implementation. In *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, pages 407–424, 2008.

[CDL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Dl-lite: Tractable description logics for ontologies. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 602–607, 2005.

[CDL+07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[CGK08] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 70–80, 2008.

[CGK13] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res. (JAIR)*, 48:115–174, 2013.

[CGL09] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 77–86, 2009.

[CGL10a] Andrea Calí, Georg Gottlob, and Thomas Lukasiewicz. Datalog extensions for tractable query answering over ontologies. *Semantic Web*

Information Management: a Model-Based Perspective, pages 249–279, 2010.

[CGL+10b] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 228–242, 2010.

[CGL12] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[CGP10a] Andrea Calì, Georg Gottlob, and Andreas Pieris. Advanced processing for ontological queries. *PVLDB*, 3(1):554–565, 2010.

[CGP10b] Andrea Calì, Georg Gottlob, and Andreas Pieris. Query answering under non-guarded rules in datalog+/-. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, pages 1–17, 2010.

[CGP11] Andrea Calì, Georg Gottlob, and Andreas Pieris. New expressive languages for ontological query answering. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011.

[CLM81] Ashok K. Chandra, Harry R. Lewis, and Johann A. Makowsky. Embedded implicational dependencies and their inference problem. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, pages 342–354, 1981.

[CM09] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.

[Cou89] Bruno Courcelle. The monadic second-order logic of graphs, II: infinite graphs of bounded width. *Mathematical Systems Theory*, 21(4):187–221, 1989.

[CV85] Ashok K. Chandra and Moshe Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. Comput.*, 14(3):671–677, 1985.

[DEGV01] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[DNR08] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 149–158, 2008.

[ELOS09] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Simkus. Query answering in description logics with transitive roles. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 759–764, 2009.

[EOS+12] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for horn-shiq plus rules. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[FKMP05] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[GHK+13] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res. (JAIR)*, 47:741–808, 2013.

[GKK+14] Georg Gottlob, Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, Thomas Schwentick, and Michael Zakharyaschev. The price of query rewriting in ontology-based data access. *Artif. Intell.*, 213:42–59, 2014.

[GMP14] Georg Gottlob, Marco Manna, and Andreas Pieris. Polynomial combined rewritings for existential rules. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, 2014.

[GMP15] Georg Gottlob, Marco Manna, and Andreas Pieris. Polynomial rewritings for linear existential rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2992–2998, 2015.

[GO13] Gösta Grahne and Adrian Onet. Anatomy of the chase. *CoRR*, abs/1303.6682, 2013.

[GPT13] Georg Gottlob, Andreas Pieris, and Lidia Tendera. Querying the guarded fragment with transitivity. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 287–298, 2013.

[HLSW15] Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. Efficient query rewriting in the description logic EL and beyond. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3034–3040, 2015.

[HS99] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Log. Comput.*, 9(3):385–410, 1999.

[KLMT15] Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo. Sound, complete and minimal ucq-rewriting for existential rules. *Semantic Web*, 6(5):451–475, 2015.

[KLT+09] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. Combined FO rewritability for conjunctive query answering in dl-lite. In *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009*, 2009.

[KLT+10] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in dl-lite. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, 2010.

[KLT+11] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2656–2661, 2011.

[KR11] Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 963–968, 2011.

[KR14] Markus Krötzsch and Sebastian Rudolph. Complexities of nominal schemas. In *Informal Proceedings of the 27th International Workshop*

on Description Logics, Vienna, Austria, July 17-20, 2014., pages 270–273, 2014.

[LMR13] Michel Leclère, Marie-Laure Mugnier, and Swan Rocher. Kiabora: An analyzer of existential rule bases. In *Web Reasoning and Rule Systems - 7th International Conference, RR 2013, Mannheim, Germany, July 27-29, 2013. Proceedings*, pages 241–246, 2013.

[LMU16] Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. On bounded positive existential rules. In *Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22-25, 2016.*, 2016.

[LR96] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *12th European Conference on Artificial Intelligence, Budapest, Hungary, August 11-16, 1996, Proceedings*, pages 323–327, 1996.

[LSTW13] Carsten Lutz, Inanç Seylan, David Toman, and Frank Wolter. The combined approach to OBDA: taming role hierarchies using filters. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 314–330, 2013.

[Mar09] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 13–22, 2009.

[MSS05] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.

[Mug11] Marie-Laure Mugnier. Ontological query answering with existential rules. In *Web Reasoning and Rule Systems - 5th International Conference, RR 2011, Galway, Ireland, August 29-30, 2011. Proceedings*, pages 2–23, 2011.

[One13] Adrian Onet. The chase procedure and its applications in data exchange. In *Data Exchange, Integration, and Streams*, pages 1–37. 2013.

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[PLC+08] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[RT14] Sebastian Rudolph and Michaël Thomazo. Mixing materialization and query rewriting for existential rules. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 897–902, 2014.

[Tho13] Michaël Thomazo. *Conjunctive Query Answering Under Existential Rules - Decidability, Complexity, and Algorithms*. PhD thesis, Montpellier 2 University, France, 2013.

[ZZY15] Heng Zhang, Yan Zhang, and Jia-Huai You. Existential rule languages with finite chase: Complexity and expressiveness. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1678–1685, 2015.

Abstract

In this thesis we investigate the issue of querying knowledge bases composed of data and general background knowledge, called an ontology. Ontological knowledge can be represented under different formalisms and we consider here a fragment of first-order logic called existential rules (also known as tuple-generating dependencies and Datalog+/-). The fundamental entailment problem at the core of this thesis asks if a conjunctive query is entailed by an existential rule knowledge base. General existential rules are highly expressive, however at the cost of undecidability. Various restrictions on sets of rules have been proposed to regain the decidability of the entailment problem. Our specific contribution is two-fold. First, we propose a new tool that allows to unify and extend most of the known existential rule classes that rely on acyclicity conditions to tame infinite forward chaining, without increasing the complexity of the acyclicity recognition. Second, we study the compatibility of known decidable rule classes with a frequently required modeling construct, namely transitivity of binary relations. We help clarifying the picture of negative and positive results on this question, and provide a technique to safely combine transitivity with one of the simplest, yet useful, decidable rule classes, namely linear rules.

Keywords: Artificial Intelligence, Knowledge Representation and Reasoning, Datalog+/-, Existential Rules, Conjunctive Queries

---

Résumé

Dans cette thèse, nous nous intéressons au problème d'interrogation de bases de connaissances composées de données et d'une ontologie, qui représente des connaissances générales sur le domaine d'application. Parmi les différents formalismes permettant de représenter les connaissances ontologiques, nous considérons ici un fragment de la logique du premier ordre appelé règles existentielles (aussi connues sous le nom de "tuple generating dependencies" et Datalog+/-). Le problème fondamental de conséquence logique au coeur de cette thèse demande si une requête conjonctive est conséquence d'une base de connaissances. Les règles existentielles étant très expressives, ce problème est indécidable. Toutefois, différentes restrictions sur les ensembles de règles ont été proposées afin d'obtenir sa décidabilité. La contribution de cette thèse est double. Premièrement, nous proposons un outil qui nous permet d'unifier puis d'étendre la plupart des classes de règles connues reposant sur des notions d'acyclicité assurant la finitude du chaînage avant. Deuxièmement, nous étudions la compatibilité des classes décidables de règles existentielles connues avec un type de connaissance souvent nécessaire dans les ontologies: la transitivité de relations binaires. Nous aidons à clarifier le paysage des résultats positifs et négatifs liés à cette question et fournissons une approche permettant de combiner la transitivité avec les règles existentielles linéaires.

Mots clefs: Intelligence Artificielle, Représentation des Connaissances et Raisonnement, Datalog+/-, Règles Existentielles, Requêtes Conjonctives