

# Rule-based Languages for Reasoning on Data : Analysis, Design and Applications

Federico Ulliana

▶ **To cite this version:**

Federico Ulliana. Rule-based Languages for Reasoning on Data : Analysis, Design and Applications. Computer Science [cs]. Montpellier University, 2021. tel-03514093

**HAL Id: tel-03514093**

**https://hal-lirmm.ccsd.cnrs.fr/tel-03514093**

Submitted on 7 Jan 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Habilitation à Diriger des Recherches

---

## Rule-Based Languages for Reasoning on Data: Analysis, Design, and Applications

---

**Federico Ulliana**

**LIRMM, Inria, Univ. Montpellier, CNRS**



# CONTENTS


**1 Research Activity** **3**

1.1 Overview . . . . . 3

1.2 The Existential Rule Framework . . . . . 5

1.3 Existential Rules Boundedness . . . . . 16

1.4 Reasoning on NoSQL Databases . . . . . 30

1.5 Applications of Ontologies and Rules . . . . . 42

**2 Research project** **48**


*Je déclare avoir respecté, dans la conception et la rédaction de ce mémoire d'HDR, les valeurs et principes d'intégrité scientifique destinés à garantir le caractère honnête et scientifiquement rigoureux de tout travail de recherche, visés à l'article L.211-2 du Code de la recherche et énoncés par la Charte nationale de déontologie des métiers de la recherche et la Charte d'intégrité scientifique de l'Université de Montpellier. Je m'engage à les promouvoir dans le cadre de mes activités futures d'encadrement de recherche.*

# 1 Research Activity

## 1.1 Overview

The intelligent exploitation of data poses the significant challenge of devising automated reasoning techniques able to improve access to information. Today, with data produced at high rates and in a large number of forms, exploiting available information becomes increasingly difficult. Nevertheless, in this dynamic and complex situation, there is a point of convergence: *domain-specific knowledge*. Data is generated by software applications, which are always proper to a specific context and domain. Hence, regardless of the many disparities, a common ground for the data produced in an application domain always exists. Formalized domain-specific knowledge gives a conceptual point of view of the domain of data, by defining the types of the objects involved, as well as the relations among them. This provides a lens through which diverse data can be looked at. It allows one to equalize databases presenting structural differences as well as to push further their value in terms of informative content. Importantly, formal knowledge can be taken into account automatically by leveraging on reasoning, in a way which is transparent for users.

Modeling and leveraging knowledge through formal languages is at the essence of the Knowledge Representation and Reasoning (KR&R) field of Artificial Intelligence. However, this also concerns the fields of Databases and the Semantic Web. In the last decade, these three domains have been actively interacting on the different facets and issues related to the intelligent exploitation of data. The questions range from the foundations of reasoning, to performance issues, to the development and deployment of Web standards and solutions. Domain-specific knowledge is typically expressed by ontologies or, more generally, by rule-based languages with a logical semantics. The purpose of rules is perhaps described at best by Gottlob: *"rules encapsulate transferable knowledge, arising from human experience, common sense, definitions, norms, laws, and regulations"* but this also includes *"knowledge that can be learned by machines"* [68]. Rules make one able to deal with intensionality and incompleteness in the data. This is what makes them able to enrich databases with novel and pertinent information thereby increasing their informative content and exploitability.

The research activity presented here revolves around rule-based languages for reasoning on data, with a particular emphasis on *existential rules*. This is an expressive formalism that encompasses well known formalisms like Datalog and Horn Description Logics. The main problem we are concerned with is that of *answering queries* under sets of existential rules. This is commonly referred as ontology mediated query answering (or theory-mediated query answering, if we see rules as logical theories). The three main topics that will be covered are the *analysis*, *design*, and *applications* of rules languages for *reasoning on data*. Our attention has been on understanding the power of logical reasoning, in particular in the light of reusing existing database technology such as NoSQL systems.

The static analysis of rule based languages serves at deriving properties that can reveal to be key for choosing the right reasoning strategy, as well as for a better reuse of rules in a collaborative setting like that of Web ontologies. The issue we have investigated is *boundedness*. This property concerns the recursivity of rules, and asks whether the depth of reasoning is independent on the input data.

This has several important implications on reasoning and enables many optimizations and modularizations of rules. Novel decidable cases and insights of the nature of the property have been derived.

The design of a rule language is fundamental to make sure that reasoning is doable provided the applicative constraints one is subject to. We have been pursuing the goal of designing families of rule languages for reasoning on top of NoSQL databases, like document oriented key-value stores which are praised for their performances. The challenge here is to comply with the constraint of the API of the underlying systems and to devise virtualization approaches based on query rewriting.

Finally, applications are very important to transfer results as well as motivating new theoretical questions. Two projects will be presented, both characterized by the goal of exploiting ontologies and rules to allow for a better exploitation of 3D models. From one side, the My Corporis Fabrica project aimed at defining an ontology for navigating through 3D models of human anatomy. From the other side, the 3D Assembly Analysis project aimed at combining a geometric analysis with reasoning for a better exploitation of CAD models. These projects show how the use of ontologies can improve the exploitation of 3D models, and at the same times outlines many reasoning features required by applicative environments.

Each topic will be developed in a dedicated section of the manuscript. Every section will aim at providing a coherent and uniform presentation of the contributions that have been accumulated through multiple investigations. To make the document more self contained, formal definitions will be given. We will occasionally take the freedom of renaming certain notions introduced in authored papers towards the goal of making things clearer. Our aim is to distill and present the keystones of the main contributions; the eager reader will be nevertheless referenced to the proper source to delve more into the technical development.

Always for concision, this document will focus primarily on the issues related to reasoning on data. So, one part of the researches carried will not be covered. This concerns the work done on the modularization of deductive triplestores, on semi-structured databases and XML, as well as a multi-disciplinary collaboration with the Center for Structural Biochemistry in Montpellier on the design of biological devices.

The manuscript is organized as follows. Section 1.2 introduces the setting of existential rules, which is the common ground for all presented contributions. Section 1.3 presents the results on static analysis and boundedness while Section 1.4 is dedicated to the design of rule base languages. Section 1.5 presents the applications of rules for reasoning of data. Finally, Section 2 presents the research project stemming by all of the research activities willing at pushing further the use of rules for exploiting data in heterogeneous and federated data-integration settings. Citations to authored work will be highlighted to ease the reading.

## 1.2 The Existential Rule Framework

### Origins and Relevance

Existential rules are positive first order logic formulas of the form $\forall X\big[Body(X,Y) \rightarrow \exists Z\ Head(X,Z)\big]$ where *Body* and *Head* are conjunctions of atoms and *Head* can have existentially quantified variables. These formulas are ubiquitous in many contexts, as they allow one to model many tasks related to data and knowledge. These rules have long been studied in the 80's in database theory as very general database constraints called Tuple Generating Dependencies (or TGDs) [24]. They also emerged as the natural logical translation of rules in Conceptual Graphs, that have been studied since the 90's [49]. In the mid 2000, they became the foundations for relational schema mappings in data exchange and integration [59]. Around 2009, with the gain of importance of ontologies, Description Logics, and Semantic Web languages, the formalism has been brought to light again by two parallel initiatives. From one side, the Datalog± family of rule languages [36]. The "+" symbolizes the extension of Datalog rules with value invention, that is, the ability to deduce the existence of unknown data values via existentially quantified variables. The "-" represents the restriction of the (very expressive) resulting language to well-behaving fragments. From the other side, existential rules have been proposed as an ontological language for KR&R [15]. Existentially quantified variables are considered a key feature in to reason in open domains, where not all individuals are supposed to be known in advance (as opposed to closed domains typically considered in databases). In both cases, one of the motivation was to have a language which can be a valid complement to Description Logics (DLs). And indeed, from the perspective of the design of a KR language, existential rules have an orthogonal approach with respect to DLs. They allow for predicates of any arity (whereas DLs employ binary relations) and complex structures (whereas DLs essentially describe treelike structures).[1] More precisely, existential rules generalize the family of Horn Description Logics and by this they also generalize Semantic Web languages and standards of the OWL family. Finally, from a purely logical perspective, sets of existential rules can be seen as Horn first-order logic theories. Their study is thus the continuation of the research for decidable fragments of first-order logic. Decidability here is intended with respect to problems like logical entailment, which is embodied in conjunctive query answering, and that can also be seen as a special case of theorem proving. In conclusion, existential rules are an important formalism for modeling and studying tasks related with data and knowledge, that furthermore touches at different domains like knowledge representation and reasoning, databases, the Semantic Web, and logics.

The remainder of this section will be dedicated to presenting the basic definitions of the existential rule framework, as well as the main approaches for reasoning with these rules: the chase and query rewriting. This will set the ground for a presentation of our research outcomes which will follow in the next sections.

[1] We refer to [97] for a recent side-by-side comparison of the two formalisms.

We mainly follow the formalization of [56] and [89] to which we shall refer to for details and examples. We place ourselves in a First-Order Logic (FO) setting, without functional symbols and equality. A relational vocabulary $(Pred, Cnst)$ is made of a finite set of predicates $Pred$ and a finite set of constants $Cnst$. We also assume a countably infinite set of variables $Vars$ used in formulas. Hereafter, we always assume a fixed vocabulary when defining objects. A *Knowledge Base* (KB) is a logical theory made by a *factbase* (holding factual knowledge on the domain of interest) and a *rulebase* (holding general knowledge on the domain of interest). A *factbase*, denoted by $F$, is an existentially quantified conjunction of atoms which is closed, i.e., every variable is existentially quantified.

In the formal development, it will be convenient to see a factbase as the set of atoms if contains. This defines a relational structure that can also be seen as a hypergraph. The set $\{p(x,y), q(y,z)\}$ for instance will represent the formula $\exists x, y, z.\ p(x,y) \wedge q(y,z)$. Then, a *substitution* is a mapping from a set of variables to a set of terms. A *homomorphism* $h$ from a set of atoms $A$ to a set of atoms $B$, denoted by $h : A \to B$, is a substitution of vars$(A)$ with terms$(B)$ such that $h(A) \subseteq B$. In this case we also say that $A$ is more general than $B$, denoted by $A \geq B$. It is known that a factbase $F$ logically entails a factbase $G$, denoted by $F \models G$, if and only if there exists a homomorphism from $G$ to $F$ seen as sets of atoms (structures) [48]. An *isomorphism* is a bijective homomorphism. A subset $F \subseteq G$ is a *retract* of $G$ if there exists a substitution $\sigma$ that is the identity on the terms of $F$ such that $\sigma(G) = F$. In this case, $\sigma$ is also called a retraction from $G$ to $F$. A factbase $F$ is a *core* if none of its strict subsets is a retract.

A *rulebase* is a set of *existential rules*, denoted by $R$, that are FO formulas of the form

$$\forall X \big[ Body(X,Y) \to \exists Z\ Head(X,Z) \big]$$

where $X, Y, Z$ are sets of variables and *Body* and *Head* conjunctions of atoms. We denote a rule by $r$. We call $Z$ the set of *existentially quantified* variables of the rule. If $Z = \emptyset$ the rule is called *datalog*. We call $X$ the set of *frontier* variables of $r$, also denoted by fr$(r)$, these are the variables that belong to both the body and the head. W.l.o.g., we assume the frontier of a rule to contain at least one variable. Universal quantification will be omitted next. Also, it will be sometimes convenient to consider a rule simply as a pair of sets of atoms $(B, H)$. Finally, we use body$(r)$ to denote $B$ and head$(r)$ to denote $H$.

An *FO-query* $q(x_1, ..., x_n)$ is a FO formula whose free variables (called answer variables) are exactly $\{x_1, ..., x_n\}$. A query which does not have any free variable is called *Boolean*. A *conjunctive query* (CQ) is an FO-query made by an existentially quantified conjunction of atoms. A *union of conjunctive queries* (UCQ) $Q$ is a set of CQs all sharing the same free variables.

An *answer* to a FO-query $q(x_1, ..., x_n)$ on a knowledge base $(F, R)$ is a substitution $\mathbf{a} : \{x_1, ..., x_n\} \to Cnst$ such that $R \cup F \models q(\mathbf{a})$, where $\models$ denotes the classical logical consequence and $q(\mathbf{a})$ is the Boolean query obtained from $q$ substituting each $x_i$ with $\mathbf{a}(x_i)$.

Among the many reasoning tasks associated with a knowledge base we focus our attention on the *conjunctive query answering* decision problem (QA hereafter), that given in input an existential rules knowledge base $(F,R)$, a conjunctive query $q$, and a substitution $\mathbf{a}$, asks to decide if $\mathbf{a}$ is an answer for $q$ on $(F,R)$. How to devise query answering algorithms that reflect the model-theoretic relations we presented? The two grand families of algorithms developed for QA can be classified as forward-chaining and backward chaining techniques. A common point to both is that they aim at reducing the QA problem to a classical database query answering (i.e., QA with an empty set of rules) by embedding the rules either into the facts or into the query.

Forward chaining is decomposed into a materialization (or saturation) step, which extends the input factbase with the inferences enabled by rules, followed by the evaluation of the query against the extended factbase. Backward chaining is decomposed into a query rewriting step, which computes a set of rewritings for the input query, followed by the evaluation of the rewritten query against the factbase (thereby leaving the data untouched). These schemes can also be combined, leading to so called combined approaches, where different portions of rules are treated with different techniques [93, 15].

Both the forward and the backward chaining approaches rely on a fixpoint operator to cope with rule semantics. Indeed, materialization should continue until the point where only redundant facts are added to the dataset, while rewriting should continue until the point where only redundant queries are added to the rewriting set. It is understood that both processes may not terminate and that entailment with existential rules is undecidable [47]. Deciding halting of the forward and backward chaining processes is undecidable as well (see e.g., [57, 15, 70, 65, 23]), and a number of sufficient syntactic conditions for termination have been proposed in the literature [101, 71, 108, 44].

To better understand the expressivity of the different existential rules fragments, these can be classified with respect to the termination of forward chaining and the backward chaining. The goal of the next section will be exactly that of presenting these strategies and defining these classes of rules. It is important to mention that [15] defines a third property ensuring decidability of OMQA, namely bounded-treewidth rulesets, which contains the Description Logic $EL$ [93] and the family of guarded rules [35, 16]. This class will not be covered here, and we refer to [97] for a recent paper presenting it.

## THE CHASE

We start by presenting a family of forward-chaining algorithms collectively called the *chase*. The chase is reasoning approach that sees rules as constraints that the factbase should be compliant with. In the case where a constraint is not satisfied, the factbase is extended with new atoms satisfying it, until fixpoint. The interest of this procedure is that, when the chase terminates, it outputs a (finite) model of the knowledge base, which moreover is universal [57]. Universal means that it can be mapped by homomorphism to any other model of the KB. This is very important, as it suffices to answer any CQ entailed by the KB, as an answer on the universal model is an answer in every model.

The main goal of this section is to present several chase variant that will be subject of our study. These are the *oblivious* chase [35], the *semi-oblivious* chase [96], the *restricted* or standard chase [60], and the *equivalent* chase [108]. For concision, we skip the presentation of the *Skolem* [96] and the *core* chase [57]. The Skolem chase coincides with the semi-oblivious chase [96] and the equivalent chase coincides with the core chase [108]. To be more precise, by "coincide" here we regard the termination of the chase, and mean that the class of rulesets terminating wrt the two chase variants are the same. For a more comprehensive survey, we refer the interested reader to [70] or [54].

All chase variants compute logically equivalent results. Nevertheless, they differ on their ability to detect logical redundancies possibly caused by the presence of fresh existentially quantified variables (we may also call *labelled nulls*, or simply nulls, using a database-oriented terminology) that are introduced by the procedure. Indeed, a factbase that contains nulls may be logically equivalent to one of its strict subsets, in which case we call it *redundant*. As deciding whether a factbase is redundant is computationally difficult (in fact, NP-complete [48]), a given chase variant may choose to detect only specific cases of redundancy. What is important is that the ability to detect redundancies has a direct impact on the termination of the chase, and only variants like the core chase or the equivalent chase terminate if and only if the knowledge base has a finite universal model.

APPLYING RULES To formally introduce the chase variants, we start from the application of rules. Let $F$ be a factbase and $r = (B, H)$ a rule. We say that $r$ is applicable on $F$ if there is a homomorphism $h$ from $B$ to $F$. Then, the pair $\mathtt{t} = (r, h)$ is called a *trigger* for $F$ and $r$. We denote by $h^{safe}$ the extension of $h$ which maps all existential variables in $H$ to fresh variables *indexed* by the trigger $\mathtt{t}$. More precisely, for each existential variable $z$ in $H$, we define $h^{safe}(z) = z_{\mathtt{t}}$. In this case, we call the factbase $F \cup h^{safe}(H)$ the *effective application* of $\mathtt{t}$ on $F$. Given a trigger $(r, h)$ we denote by $h_{|\mathrm{fr}(r)}$ the restriction of $h$ to the frontier variables of $r$.

Clearly, two equal triggers produce equal results, and thus the repetitive application of the same trigger will not be considered by the basic chase variants, the oblivious chase. Also, triggers for the same rule $\mathtt{t}_1 = (r, h_1)$ and $\mathtt{t}_2 = (r, h_2)$ produce isomorphic results if they agree on the image of the frontier variables, that is, $h_{1|\mathrm{fr}(r)} = h_{2|\mathrm{fr}(r)}$. In this case we denote it by $\mathtt{t}_1 \sim \mathtt{t}_2$; we will employ this notation later when presenting the semi-oblivious chase. We are now ready to give a basic notion of *sequential* derivation.

**Definition 1.** *A derivation* $\mathsf{D}$ *from* $F$ *and* $R$ *is a (possibly infinite) sequence of triggers* $\mathtt{t}_1, \mathtt{t}_2, \ldots$ *producing a sequence of factabases* $(F =) F_0, F_1 \ldots$ *where every* $F_{i>0}$ *is the effective application of* $\mathtt{t}_i$ *on* $F_{i-1}$.

*We denote by* $F\langle\mathsf{D}\rangle$ *the factbase resulting from chasing* $F$ *with* $\mathsf{D}$*, which is defined as* $F\langle\mathsf{D}\rangle = \bigcup_{i \geq 0} F_i$.

This definition sets the backbone of any sequential derivation by just ensuring that the sequence of triggers is cohesive. This means that triggers can indeed be applied starting from the initial factbase one after the other. As a complement of notation, we denote by $\mathsf{D}_i$ the prefix $\mathtt{t}_1, \ldots, \mathtt{t}_i$ of $\mathsf{D}$.

From our perspective, we see a chase variant simply by the set of derivations it enables. These are defined by basically playing on the basis of three dimensions.

- The first one is what we call here the *compliance condition* (also called applicability condition) which given a trigger which is applicable at a certain point of the derivation, it either promotes the trigger among the applied ones or discards it as deemed redundant. This defines most of the capacity of the chase at terminating by avoiding redundancies, but it is not the only one.
- The second one is the *prioritization* on triggers. Arbitrary derivations like those just defined impose no priority on triggers. However, prioritizations schemes implementing breadth-first strategies [56], or applying datalog rules first [87] can have an impact on termination for chase variants using the compliance condition of the restricted chase [54].
- The third one is the *scope* of the verification of the compliance condition. In sequential derivations, the compliance condition is be verified with respect to all previously applied triggers. In other strategies like the parallel [57] or the 1-parallel [64, 25] just a subset of the triggers is considered. This again can have an impact on termination [54].

These aspects may interact in some non-obvious way, and give raise to a family of chase procedures where every variant halts differently.

COMPLIANCE As already said, the *compliance condition* is what makes that a trigger (that *could* be applied) is actually applied at a certain point of the derivation. It is interesting to remark the nature of the compliance condition. For some chases, this may be defined depending on the "syntactical" components of the triggers that have been formerly applied. This is the case for example of the oblivious and the semi-oblivious chase. For other chases like the restricted and the equivalent chase, this condition rather depend on the *inferences* that have been previously made, that is, what the triggers produced rather than how the triggers looked like before being applied. It should not be surprising then that these chases will be better at detecting redundancies. Let us define the compliance condition for our chase variants. We refer to Example 7 of [56] for an illustration of these conditions.

**Definition 2.** *Let* $\mathsf{D}$ *be a finite derivation, from* $F$ *and* $R$*, and* $\mathsf{t}$ *a trigger which is applicable on* $F\langle\mathsf{D}\rangle$*.*

*Then, with respect to* $\mathsf{D}$*, the trigger* $\mathsf{t}$ *is said:*

- obl-compliant *if* $\mathsf{t}$ *does not belong to* $\mathsf{D}$
- sob-compliant *if there is no* $\mathsf{t}' \in \mathsf{D}$ *such that* $\mathsf{t} \sim \mathsf{t}'$

*Further, let* $F'$ *be the effective application of* $\mathsf{t}$ *on* $F\langle\mathsf{D}\rangle$*. Then, with respect to* $\mathsf{D}$*, the trigger* $\mathsf{t}$ *is said:*

- res-compliant *if there is no retraction from* $F'$ *to* $F\langle\mathsf{D}\rangle$
- eqv-compliant *if there is no homomorphism from* $F'$ *to* $F\langle\mathsf{D}\rangle$

*An* $X$-derivation *is a derivation where every trigger* $\mathsf{t}_{i+1}$ *is* $X$*-compliant wrt the prefix* $\mathsf{D}_i$*.*

*The class of all the* $X$*-derivations is the* $X$*-chase.*

Note that obl-compliancy (for the oblivious chase) simply excludes the multiple applications of the same trigger. Then, sob-compliancy (for the semi-oblivious chase) will avoid the application of two triggers producing isomorphic results. The res-compliancy (for the restricted chase) checks if the atoms produced by a trigger can be folded back by retraction. This means that the test is local

with respect to where the rule has been applied, as the image of frontier variables cannot change by retraction. Finally the eqv-compliancy (for the equivalent chase) checks if the atoms produced by a trigger can be folded back, this time globally, with respect to the whole factbase. This last one is the most expressive condition one can have, and it is the only one able to ensure that the chase outputs a finite universal model of the knowledge base if and only if this one exists.

As already said, the different compliance condition for triggers set the main capacity of a chase variant at handling redundancies, and the less redundancy a chase produces, the more it terminates. The last bit we need to precisely define the notion of terminating derivation is *fairness*. Simply put, fairness ensures that no relevant trigger has been forgotten. Without fairness, it would be possible to halt a derivation by skipping some constraints. However, this would not guarantee a universal model of the knowledge base. Note that fair derivations may be infinite. However, finite fair derivations produce (finite) universal models of the knowledge base.

**Definition 3.** *An X-derivation* $\mathsf{D}$ *is* fair *if for every trigger* $\mathsf{t}$ *which is X-compliant with the prefix* $\mathsf{D}_i$ *there is a* $k > i$ *such that one of the following holds.*

- $\mathsf{t}_k = \mathsf{t}$
- $\mathsf{t}$ *is not X-compliant with the prefix* $\mathsf{D}_k$

*An X-derivation is* terminating *if it is both fair and finite.*

In words, a derivation is fair if, for every trigger applicable on one of its prefixes, either the trigger has been applied or it became redundant later.

PRIORITIZATION The second important dimension in a derivation is the order in which rules are applied. This aspect is often blurred by the formal study but it emerges immediately when effectively implementing the procedure. The first prioritization we can think of is the breadth-first one, where rules are applied rank by rank. This is intuitive, all applicable rules are applied on the initial factbase (at rank 0) thereby yielding a new factbase (at rank 1) where the process repeats until fixpoint. But are breadth-first derivations always the optimal choice for achieving termination? This depends on the chase. The answer is positive for variants like the oblivious, semi-oblivious chase, and the equivalent. The answer is negative for the restricted chase.

Another prioritization that has been proposed more recently is *datalog-first* [87]. In this scheme, the idea is to produce the most possible knowledge concerning *only* the terms of the current factbase through datalog rules, before introducing new (possibly redundant) nulls. Again, for the restricted chase, this can have an impact on termination.

Among all prioritization strategies, we focus our attention on *sequential breadth-first derivations*. The reason is twofold. First, they will be subject of study in the section on boundedness. Second, we also want to clarify this notion and stress the differences with *parallel* breadth-first derivations. Towards this aim, we need first to define the notion of rank of an atom (or a trigger).

**Definition 4.** *Let* $\mathsf{D}$ *be a derivation from* $F$ *and* $R$*. The rank of an atom* $\alpha$ *belonging to* $F\langle\mathsf{D}\rangle$ *is defined as*

$$\mathsf{rank}(\alpha) = \begin{cases} 0 & \text{if } \alpha \in F \\ 1 + \max\{\, \mathsf{rank}(\beta) \mid \beta \in h(B) \,\} & \text{otherwise} \end{cases}$$

*assuming that* $\mathsf{t} = \big((B,H),h\big)$ *is the* first *trigger of* $\mathsf{D}$ *producing* $\alpha$*.*

*This notion is naturally extended to triggers* $\mathsf{rank}(\mathsf{t}) = 1 + \max\{\, \mathsf{rank}(\beta) \mid \beta \in h(B) \,\}$*.*

*Finally, the* depth *of a derivation* $\mathsf{D}$ *is the maximum rank of its atom, if it is finite, and else infinite.*

Informally speaking, the atom rank does not indicate the number of triggers fired before producing it but rather the number of breadth-first steps that are needed to produce it. The notion of rank stems from breadth-first derivations but applies to any derivation. It should not be surprising that two derivations may produce the same atom at different ranks if they choose different prioritizations for their triggers, and this already occurs for datalog rules (see Example 6 of [56]). We are now ready to define breadth-first derivations.

**Definition 5.** *An* $X$*-derivation* $\mathsf{D}$ *is breadth-first if for all two consecutive triggers* $(\mathsf{t}_i, \mathsf{t}_{i+1})$ *holds that:*

- $\mathsf{rank}(\mathsf{t}_i) \leq \mathsf{rank}(\mathsf{t}_{i+1})$ *(rank-compatibility)*
- $\mathsf{rank}(\mathsf{t}_i) < \mathsf{rank}(\mathsf{t}_{i+1})$ *implies there is no trigger* $\mathsf{t}$ *that is* $X$*-compliant with* $\mathsf{D}_i$ *s.t.* $\mathsf{rank}(\mathsf{t}) = \mathsf{rank}(\mathsf{t}_i)$ *(rank-exhaustiveness)*

*The class of all the breadth-first* $X$*-derivations is the bf-*$X$*-chase.*

As expected, a breadth-first sequential derivation applies all triggers of a rank before moving to the next rank. It is well known that the oblivious and semi-oblivious chase variants are order-insensitive with respect to the application of the triggers [70]. Hence choosing a particular order in the derivations does not have an impact termination, although the ranks of atoms may fluctuate [56]. The equivalent chase is also insensitive to termination because it is able to verify logical equivalence [56].

The case of the restricted chase is different, and there are several scenarios. It may be that none of the breadth-first derivations are terminating, but a terminating derivation exists. It may also happen that some breadth-first derivations terminate while others dont. It may also happen that all breadth-first derivations terminate but there is a fair non-breadth-first derivation that does not terminate. The following example, which reprises Example 20 of [56], illustrates the three situations.

**Example 6.** *Consider the rules*

$$\begin{array}{llll} r_{p\,next_p}: & p(x,y) \to \exists z.\, p(y,z) & r_{p\,loop\text{-}second_p}: & p(x,y) \to p(y,y) \\ r_{p\,next_q}: & p(x,y) \to \exists z.\, q(y,z) & r_{q\,loop\text{-}first_p}: & q(x,y) \to p(x,x) \\ r_{q\,side_p}: & q(x,y) \to \exists z.\, p(x,z) & & \end{array}$$

*and the rulesets*

$$R_1 : \{r_{p\,next_p}, r_{p\,next_q}, r_{q\,loop\text{-}first_p}\} \quad R_2 : \{r_{p\,next_p}, r_{p\,loop\text{-}second_p}\} \quad R_3 : \{r_{p\,next_q}, r_{q\,side_p}, r_{p\,loop\text{-}second_p}\}$$

*Fixed the factbase $F = \{p(a,b)\}$, there is a terminating restricted chase for each rulebase $R_i$. However:*

- *For the KB $(F, R_1)$ no breadth-first derivation is terminating. Forcing the application of $r_{p\,next_p}$ at each rank creates an infinite path. However, a terminating derivation can be achieved simply applying $r_{p\,next_q}$ followed by $r_{q\,loop\text{-}first_p}$. This however does not satisfy exhaustiveness at rank 1.*
- *For the KB $(F, R_2)$ not all breadth-first derivations are terminating. This depends on the order within a single rank. A fair but infinite restricted derivation can be built by prioritizing $r_{p\,next_p}$ over $r_{p\,loop\text{-}second_p}$. However, if the opposite is done, the derivation terminates.* [2]
- *For the KB $(F, R_3)$, all breadth-first derivations are terminating. The application of $r_{p\,loop\text{-}second_p}$ deactivates all triggers for $r_{q\,side_p}$ at the next rank. However, a fair but infinite restricted derivation can be build by repeatedly applying $r_{p\,next_q}$ and $r_{q\,side_p}$ before $r_{p\,loop\text{-}second_p}$ of previous ranks.*

Hence, in the case of the restricted chase, breadth-first derivations are not necessarily derivations that terminate more often. As a complement of information, we mention that it can actually be shown that rank-compatible restricted chase derivations (which do not require rank exhaustiveness) are the ones who terminate sooner [56]. Finally, let us point out a property of breadth-first derivations on which build the results on the study of boundedness for breadth-first chase variants presented next; this concerns the minimality of ranks.

**Proposition 7** ([56])**.** *For a fixed knowledge base, every breadth-first derivation for the oblivious, semi-oblivious and restricted chase, produce atoms at their minimal rank.*

Since nulls are indexed by triggers (recall the definition of rule application), terms and atoms are uniquely defined among the class of derivations for a fixed knowledge base. This also implies that the breadth-first oblivious, semi-oblivious and restricted chase agree on the rank of atoms they produce.

SCOPE The third dimension for a chase is the *scope* of the verification of the compliance condition. As stated by Definition 2, sequential derivations verify the compliance condition with respect to all the applied triggers. In contrast, so called *parallel* strategies verify the compliance condition with respect to a *subset* of the applied triggers. The *parallel-breadth-first* strategy proposed the *parallel chase* [57] (and of course for Datalog [10]) is perhaps the most intuitive one. The idea is that, at each rank, all possible rule applications are triggered in parallel. Therefore, at each step, the compliance condition is verified only upon the factbase obtained at the *previous* breadth-first step. The result of all compliant triggers is then added to the current factbase. Note also that the parallel derivation is unique, as every derivation step contains a maximal set of triggers.

The *1-rule-parallel* strategy regroups all triggers of the same rank *and* the same rule. Although this could seem somewhat involved, it is the most natural approach for chase implementations that runs on top of a data management system. The rule body becomes a query for the underlying database, and from its result stem a set of triggers that can be applied in parallel [64].

[2] Note that $r_{p\,loop\text{-}second_p}$ is a datalog rule. So, this example also illustrates a case where the datalog-first strategy is more efficient than arbitrary or breadth-first derivations, as it ensures termination is all cases.

The parallel-breadth-first strategy is the way in which the saturation of Datalog programs is defined, and this is perfectly fine because there are no existential variables in datalog rules. This way of seeing the oblivious and semi-oblivious chase is also fine as these chases are order independent. For the first, we just have to avoid to apply a trigger twice (and parallel derivations ensure that). For the second, a parallel derivation may only produce several isomorphic copies of the same facts, which differ only for the trigger indexing the nulls. In reason of this both chases can be defined and studied with breadth-first-parallel derivations [28]. Combined with Proposition 7, we also have that these derivations lead to termination sooner, that is, every derivation with a minimal rank is a breadth-first one. Again, the equivalent chase is also insensitive to parallel applications, and the core chase is actually defined by first running a parallel-breadth-first restricted chase and then computing a core of the resulting factbase [57].

For the restricted chase things are again different. The reason is that in a sequential breadth-first derivation there may be some trigger deactivating other triggers of the same rank. By restricting the scope of the compliance condition, this deactivation may not be possible anymore. To illustrate, let us consider again Example 6. While there is a restricted breadth-first terminating sequence for $(F, R_2)$, the parallel-breadth-first derivation here is infinite. It can also be shown that whenever the parallel-breadth-first restricted chase terminates the sequential-breadth-first chase does [54].

At this point, the difference between a *sequential* breadth-first and a *parallel* breadth-first should be more clear at this point. For concision, we will not try to formalize parallel derivations here, and rather refer to [54] for an attempt of formalizing a number of chase variants including the parallel ones.

CHASE TERMINATION CLASSES To conclude this section we recall the definition of some important classes of rules for which the chase terminates. We focus mostly on compliance and prioritization. We always consider the *all instance* termination problem for a chase, that is, whether a given chase variant terminates on all input instances [70].

**Definition 8.** *A ruleset $R$ belongs to the class*

- $\mathsf{CT}^X_\forall$ *if for all factbase $F$ every fair $X$-derivation with the rules of $R$ is finite*
- $\mathsf{CT}^X_\exists$ *if for all factbase $F$ there is a terminating $X$-derivation with the rules of $R$*

Let us recall some known results. Holds $\mathsf{CT}^X_\forall = \mathsf{CT}^X_\exists$ for $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{eqv}\}$, but $\mathsf{CT}^{\mathsf{res}}_\forall \subset \mathsf{CT}^{\mathsf{res}}_\exists$ [70]. From the compliance conditions it also follows that $\mathsf{CT}^{\mathsf{obl}}_\forall \subset \mathsf{CT}^{\mathsf{sob}}_\forall \subset \mathsf{CT}^{\mathsf{res}}_\forall \subset \mathsf{CT}^{\mathsf{res}}_\exists \subset \mathsf{CT}^{\mathsf{eqv}}_\forall$ [70]. We have just recalled that the situation is similar for breadth-first derivation and hence we have $\mathsf{CT}^X_\forall = \mathsf{CT}^{X_{\mathsf{bf}}}_\forall = \mathsf{CT}^{X_{\mathsf{bf}}}_\exists = \mathsf{CT}^X_\exists$ for $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{eqv}\}$. However, $\mathsf{CT}^{\mathsf{res}}_\forall \subset \mathsf{CT}^{\mathsf{res}_{\mathsf{bf}}}_\forall \subset \mathsf{CT}^{\mathsf{res}_{\mathsf{bf}}}_\exists \subset \mathsf{CT}^{\mathsf{res}}_\exists$. The rulesets enjoying parallel breadth-first restricted termination are strictly between $\mathsf{CT}^{\mathsf{res}}_\forall$ and $\mathsf{CT}^{\mathsf{res}_{\mathsf{bf}}}_\forall$ [54]. Rulesets that terminate with the datalog-first strategy applied to the restricted chase are strictly between $\mathsf{CT}^{\mathsf{res}}_\forall$ and $\mathsf{CT}^{\mathsf{eqv}}_\forall$ [87].

Later when talking about boundedness, we will see that these relationships are interestingly different. In particular, going breadth-first plays an role in achieving bounded termination. Finally, it is

worth mentioning that the class $\mathsf{CT}^{\mathsf{eqv}}_{\exists}$ is equivalent to the Finite Expansion Set (FES) class of existential rules [15]. This is the largest class of rulesets where the chase terminates, we denote by $\mathsf{CT}$, and coincide with the rulesets always leading to a finite universal model of the knowledge base.

## Query Rewriting

Query rewriting is a backward chaining method for answering queries, which involves rewriting the input query using the rules. The goal of this section is to present the notion of query rewriting as well as defining the class of rulesets that admit a rewriting into a FO-query.

The key operation in query rewriting is the unification of (subsets of) the query with (subsets of) the rule head, which requires particular care due to the presence of existential variables in the rules. The notion of piece-unifier has been introduced for this purpose [15]. To present piece-unifiers, we need to pose first the notion of separating variables of a query subset. Given a subquery $q' \subseteq q$, we call *separating* the variables of $q'$ that occur in both $q'$ and $(q \setminus q')$. The definition of piece-unifier ensures in particular that, being $q'$ the subquery of $q$ unified with the head of a rule, only variables from $q'$ which are *not* separating can be unified with existential variables. Let $q$ be a query and $r = (B, H)$ a rule. A *piece-unifier* of $q$ with $r$ is a triple $\mu = (q', H', u)$ where $q' \neq \emptyset$, $q' \subseteq q$, $H' \subseteq H$, and $u$ is a substitution of $T = \text{terms}(q') \cup \text{terms}(H')$ by $T$ such that $(i)$ $u(q') = u(H')$ and $(ii)$ for all existential variable $x \in \text{vars}(H')$ and $t \in T$, with $t \neq x$, if $u(x) = u(t)$, then $t$ is neither a separating variable nor an answer variable of $q'$.

**Definition 9.** *Given a query $q$, a rule $r = (B, H)$ and a piece-unifier $\mu = (q', H', u)$ the* direct rewriting *of $q$ with $(r,\mu)$ is $\left(u^{\text{safe}}(B) \cup u(q \setminus q')\right)$, where $u^{\text{safe}}$ is an extension of $u$ substituting each non-frontier variables of $B$ with fresh variables.*

*A* rewriting sequence *is a sequence of queries $q_0, \ldots, q_k$ such that $q_{i>0}$ is a direct-rewriting of $q_{i-1}$. We say that $q'$ is a rewriting of $q$ if there is a rewriting sequence from $q$ to $q'$.*

The soundness and completeness of query rewriting has been shown in [15, 84]. Fixed a ruleset $R$, a set of rewritings $Q$ of $q$ is said to be *sound* and *complete* if, for any factbase $F$, it holds that $FR \models q$ if and only if there is $q' \in Q$ such that $F \models q'$. A ruleset $R$ is said to be First-Order Rewritable if it admits a *finite* sound and complete set of rewritings [40]. In this case, the set $Q$ be seen as a UCQ, so the previous condition becomes $F, R \models q$ if and only if $F \models Q$. It has been observed already in [113] that FO-rewritability is equivalent with the independently proposed notions of Finite Unification Set [15] and Bounded-depth Derivation Property (BDDP) [36]. In the following sections we will use the notation $\mathsf{FO\text{-}R}^{\mathscr{C}}$ to denote the rulesets that are FO-rewritable with respect to a class of conjunctive queries $\mathscr{C}$, and simply $\mathsf{FO\text{-}R}$ when the ruleset is rewritable with respect to all conjunctive queries.

A Powerful Notion To compute an FO-rewriting of a query, a (parallel) breadth-first query rewriting operator is has been given in [15]. The FUS class is precisely defined as the rulesets on which this operator terminates on all input queries. Starting from a given query $q$, all possible *direct* rewritings

are computed in parallel, level by level, until fixpoint. The rewritings produced *up to* level $i$ are denoted by $Q^i$, and of course $Q^0 = \{q\}$. The process ends if and only if the input query is FO-rewritable. At each rewriting level, it is checked whether the obtained rewritings logically entail the already computed ones. This holds if and only if $Q^{i+1} \models Q^i$.[3] In this case, the process terminates as no useful rewriting is produced at step $i$.

Visually, the rewriting process produces a tree where *i*) the root is the initial query and *ii*) the children of every node are its direct rewritings. During this process, every rewriting is compared with all other rewritings so as to identify and keep only the most general ones. We therefore look for homomorphisms across the different *paths* of the rewriting tree.

By stretching a bit the concept, we could make an analogy with the parallel breadth-first equivalent chase where at each rank $k$ the logical equivalence of the factbase obtained at rank $k+1$ with the factbase obtained at rank $k$ is verified. However, while weaker forms of redundancy have been defined for the chase, by introducing different compliance conditions, and many syntactic restrictions leading to FO-rewritability exists, there is no stronger notion than FO-rewritability based on query rewriting that we are aware of. As also remarked in [102], this suggests that the classes of rulesets enjoying FO-rewritability may not have been fully explored, despite this notion being present in a bulk part of the literature on the subject. Are the two main classes of FO-rewritable existential rules, namely *linear* and *sticky*, perhaps enjoying a stronger notion of rewritability ? Let us introduce a novel class of rules that we shall use later to formulate some conjecture for future work.

**Definition 10.** *A ruleset R belongs to the* rewriting sequence termination *class* RT *if for every rewriting sequence* $q_0, q_1, \ldots$ *producing a set of queries* $Q = \{q_0, q_1, \ldots\}$ *there is a finite subset* $Q' \subseteq Q$ *s.t.* $Q \models Q'$.

Going back to the tree representation of the rewriting set of query, computing the set of reformulations of a query for a ruleset in RT boils down to verify logical entailment of direct rewritings along a *path* of the tree, rather than the whole tree. This means that every rewriting sequence naturally terminates because it is not able to produce any more general query. We will illustrate in a moment (Example 11) a ruleset which belongs to FO-R but does not belong to RT. This is a stronger property than FO-rewritability, and we claim that it is enjoyed for instance by backward-shy rules [118], which include notable sets of rules like *linear* [36] and *sticky* sets [38]. Indeed, as outlined by [118] (Property 4), backward shy rules make that every query $q$ has a finite number of rewritings (depending on $q$ and $R$) which are not equivalent up to isomorphism. Thus, every infinite rewriting sequence with backward-shy rules must be equivalent to one of its subsets.

We will not dig further in the details of query rewriting and refer to [15] for a presentation of the notion (actually in the light of FUS sets of rules). Overall, this introductory section provided us the basic definitions existential rules, chase termination and FO-rewritability. We can now pass to the static analysis and design of rule languages issues, which will build on the notions just defined.

[3] Note that since $Q^i \subseteq Q^{i+1}$ follows $Q^i \models Q^{i+1}$ and hence the rewriting stops whenever $Q^i \equiv Q^{i+1}$

## 1.3 Existential Rules Boundedness

What is Boundedness ? Boundedness emerged during the early study of deductive databases [94, 114, 78, 53, 99] motivated by the issue of characterizing Datalog programs for which the *depth*[4] of a derivation was independent from the input data.[5] As we shall discuss later, this opens the opportunity for optimizations and more generally for a better reuse of rulebases holding ontologies. Non-recursive rulesets are clearly bounded. However, there are also recursive programs whose depth of recursion does not depend on the input data. To illustrate, consider the canonical example from [99].

$$r_1: \quad trendy(x) \wedge buys(z,y) \rightarrow buys(x,y)$$
$$r_2: \quad likes(x,y) \rightarrow buys(x,y)$$

The ruleset $R = \{r_1, r_2\}$ is recursive, because $r_1$ is (it has the same predicate in the body and the head). Yet, for every input factbase $F$, 2 parallel breadth-first rule applications always suffice to saturate $F$. By looking closer at $r_1$, we can see that its body atoms are disconnected, and this rule is filling the *buys* relation with a cartesian product between all matches of variables $x$ (trendy) and $y$ (items). This can be computed in a single breadth-first step by $r_1$ and, clearly, it does not create any new item "to buy". So to reach a fixpoint, it suffices that all items will be present in *buys* atom. For this, we just need that $r_2$ is executed. Therefore, independently from the input database, the relation *buys* can be computed in 2 parallel breadth-first steps and even 1 breadth-first sequential step if we start from $r_2$. In this case, we say that $R$ is bounded, with bound 2.

From Datalog to Existential Rules A study of boundedness for existential rules needs however a slightly refined notion than that typically used for Datalog. This should take into consideration the chase variant which is considered, for two reasons. First, as for the chase termination problem, every chase have different terminating conditions depending on its ability at filtering redundancies. Second, a chase may not impose a prioritization strategy for derivations. From this stems the need to consider whether one is interested on boundedness for *all* or *some* derivations. The following example, illustrates a bounded set of Datalog rules, for which however not all derivations are of bounded depth.

**Example 11** ([28])**.** *Consider the set $R = \{r_1, r_2\}$ made by the following rules*

$$r_1: \quad p(x,y) \wedge p(y,z) \rightarrow p(x,z) \qquad r_2: \quad p(x,y) \wedge p(u,z) \rightarrow p(x,z)$$

*This ruleset is recursive, because both $r_1$ and $r_2$ are. However, $r_2$ (whose body atoms are disconnected)*

[4] in the precise sense of Definition 4

[5] The fact that Datalog is a query language with extensional database predicates (EDB), intensional database predicated (IDB), and a special IDB predicate, we call *goal*, designated to hold the query answers, led to three flavours of the property. *Predicate* boundedness considers the fact that EDBs are the input and the predicate *goal* is the output. *Program* boundedness considers the fact that EDBs are the input and all IDBs are the output. *Uniform* (or *strong*) boundedness considers the all EDBs and IDBs predicates are both input and output. Uniform boundedness is the property we consider and refer as boundedness. Indeed, the other notions are less adapted to our study of existential rules, which do not have by default the EDB/IDB distinction nor a goal predicate.

*computes in a single step a superset of what is computed by the transitivity rule $r_1$ in many steps. So, for every factbase $F$, a single breadth-first application of rules will allow to reach a fixpoint. However, if another strategy is preferred, things can be different. Consider a* depth-*first derivations where $r_1$ is applied before applying $r_2$. In this case, for every $k \geq 0$ one can find a factbase $F$ (actually a path on $p$ of length $2^k$) and a depth-first derivation of depth $k$.*

We are now ready to define boundedness for the different chase variants.

**Definition 12.** *Let $X$ be a chase variant and $k \geq 0$ an integer. Then, we say that a ruleset $R$ belongs to*

- $\mathsf{BN}[k]^X_\exists$ *if, for all factbase $F$, there exists a fair $X$-derivation from $F$ and $R$ of depth bounded by $k$* (exists boundedness)
- $\mathsf{BN}[k]^X_\forall$ *if, for all factbase $F$, all fair $X$-derivations from $F$ and $R$ are of depth bounded by $k$* (forall boundedness)

*The class of bounded rulesets for a chase variant $X$ is defined as $\mathsf{BN}^X_\star = \bigcup_{k\geq 0} \mathsf{BN}[k]^X_\star$ with $\star \in \{\exists, \forall\}$. Finally, we denote by $\mathsf{BN}$ the class of rulesets that belong to some $\mathsf{BN}^X_\star$.*

To illustrate the definition, consider again Example 11. We have that $R \in \mathsf{BN}$ and more precisely $R \in \mathsf{BN}[2]^X_\exists$. This holds for every chase variant $X$ because no existential variable is present in $R$. Nevertheless, for all chase variant $X$ that does not impose a breadth-first strategy, $R \notin \mathsf{BN}^X_\forall$ holds because $R \notin \mathsf{BN}[k]^X_\forall$ for all $k \geq 0$.

We just defined boundedness in terms of terminating (i.e., fair and finite) derivations. It goes without saying that bounded rulesets are also terminating, i.e., $\mathsf{BN} \subset \mathsf{CT}$, and more precisely, $\mathsf{BN}^X_\star \subset \mathsf{CT}^X_\star$ for $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{res}, \mathsf{eqv}\}$. It is also not difficult to see that $\mathsf{BN}^X_\star \subset \mathsf{BN}^Y_\star$ when $\mathsf{CT}^X_\star \subset \mathsf{CT}^Y_\star$. For general existential rules, we have that $\mathsf{BN} \subset$ FO-R, as the property also induces an uniform bound on the number of parallel breadth-first rewriting steps needed to reformulate any query [89]. Very interestingly, for datalog rules, we have that $\mathsf{BN} =$ FO-R [12].

Example 11 only uses datalog rules and so also shows that $\mathsf{BN}^X_\forall \subset \mathsf{BN}^X_\exists$ for every chase variant $X$. A closer inspection reveals that the picture is rather $\mathsf{BN}^X_\forall \subset \mathsf{BN}^{X_{\mathsf{bf}}}_\forall \subseteq \mathsf{BN}^{X_{\mathsf{bf}}}_\exists \subseteq \mathsf{BN}^X_\exists$. What boundedness is pointing us here is a more fine grained inspection on reasoning which seeks at minimizing the number of rule applications to compute universal models. The property suggests that the use of breadth-first strategies could be important for termination for some chase variants. As outlined by Proposition 7, for the oblivious, semi-oblivious and restricted chase, breadth-first derivation produce atoms at their minimal rank [56] and for the two former variants this lead to termination sooner. Although the same property does not hold for the equivalent chase (see Example 18 of [56]), for this variant again breadth-first derivations form a valid strategy. Overall, this translates to the fact that for some variants studying breadth-first derivations coincides with studying existential boundedness.

**Proposition 13** ([56])**.** $\mathsf{BN}^{X_{\mathsf{bf}}}_\forall = \mathsf{BN}^{X_{\mathsf{bf}}}_\exists = \mathsf{BN}^X_\exists$ *for* $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{eqv}\}$ *while* $\mathsf{BN}^{\mathsf{res}_{\mathsf{bf}}}_\forall \subset \mathsf{BN}^{\mathsf{res}_{\mathsf{bf}}}_\exists \subset \mathsf{BN}^{\mathsf{res}}_\exists$

Finally, note that $\mathsf{BN} = \mathsf{BN}^{\mathsf{eqv}}_\exists$. This can be seen as the largest class of bounded rulesets for which a (finite) universal model of the knowledge base can always be computed in a predefined number of steps, independently from the input factbase.

OPEN QUESTIONS Deciding if an arbitrary set of existential rules belongs to any $\mathsf{BN}^X_\star$ is an undecidable problem. This immediately follows from the undecidability of Datalog (uniform) boundedness [76, 95]. Nevertheless, this negative result does not tell us much about the decidability of the problem for other fragments of existential rules (that do not include datalog rules). The same holds for the problem of deciding whether a ruleset belongs to $\mathsf{BN}[k]^X_\star$, also called $k$-boundedness (note here the bound $k$ is part of the question). Finally, it is interesting to dig more on the property, and understand its relations with other known properties such as chase termination and FO-rewritability. All of these questions have been the subject of our study, which will be presented in the following sections.

WHY IS BOUNDEDNESS IMPORTANT ? As already outlined, the property implies that the ruleset behaves in a non-recursive way, although syntactic conditions may fail to capture this. Being able to recognize boundedness is important in three main areas, namely the *optimization* of reasoning tasks and the *static analysis* and *reusability* of the rulebase.

Boundedness impacts the termination of many algorithms used for conjunctive query answering. At a very high level, the idea is that if a ruleset is bounded by $k$ then forward chaining procedure can halt after $k$ parallel breadth-first steps thereby avoiding the verification of any terminating condition.

Boundedness has been recently identified as a key properties for computing *trigger graphs* that are data structures guiding and optimizing the forward chaining reasoning process [120]. It has been shown in particular that a rulebase admits a trigger graph if and only if it belongs to $\mathsf{BN}$.

Boundedness ensures first order rewritability. Also, a proper conjunctive query rewriting operator can be designed in a such a way that it terminates within $k$ breadth-first steps, with $k$ the bound for the forward chaining [89]. In turn, this gives the decidability of conjunctive query containment under existential rules, which, given two conjunctive queries, asks if the set of answers to the first query is included in the set of answers to the second query, for any factbase.

Finally, recognizing boundedness is important for the reusability of rulebases set via techniques like forgetting [124]. And indeed, one of the main motivations for the study realized on this problem comes from a related problem raised in [112].

THE JOURNEY I first got interested by boundedness during my postdoctoral studies when looking at the problem of modularization for RDF Triplestores [112]. After joining GraphIK, I then start working on the subject for existential rules with Michel Leclère and Marie-Laure Mugnier team since 2015. All the presented contributions are co-authored with them. After some preliminary work on the topic [89] our research project on boundedness ramified in three paths. In the context of the thesis of Stathis Delivorais that we co-supervised we focused on the study of the $k$ boundedness property covering a very large spectrum of chase variants [55, 56] that Stathis even further extended in his thesis [54]. In the context of a collaboration with Michaël Thomazo we then focused on the case of linear rules (for which chase termination and boundedness coincide) [88]. Finally, in a collaboration with Pierre Bourhis and Sophie Tison, we worked on the characterization of boundedness in terms of known properties of rulesets[28]. Beside technical contributions, this line of research opened the

investigations for boundedness in existential rules. The topic is gaining momentum [124, 120, 102] and, as we shall discuss, several interesting questions remain to be answered ranging from the characterization and decidability of boundedness to a deeper understanding of first-order rewritability [102].

### DECIDABILITY OF $k$-BOUNDEDNESS

In the PhD Thesis of Stathis Delivorias we focussed our attention on the $\forall$-$k$-boundedness problem, that is, the problem of recognizing if a set of rules belongs to $\mathsf{BN}[k]^X_\forall$ [56, 55]. This issue has also been touched in [28]. The interest of the problem comes from the fact that, being boundedness generally undecidable for existential rules, the $\forall$-$k$-boundedness property can reveal to be a practically useful approximation, that allows one to guess and verify a (typically small) bound for a set of rules. The problem has been studied for the chase variants presented in Section 1.2, as well as for their breadth-first version. To the best of our knowledge, this has been the first study on $k$-boundedness. Independently, the problem has also been considered in [63].

CONTRIBUTION At a very high level, the main contribution of this work can be resumed by the following theorem.

**Theorem 14** ([56])**.** *Membership to* $\mathsf{BN}[k]^X_\forall$ *is always decidable for the oblivious, semi-oblivious, and restricted chase, as well as their breadth-first variants.*

It is worth noting that a general approach to derive decidability results for generally undecidable static analysis problems like boundedness is to restrict the rule language. Here, by focusing on the $\forall$-$k$-boundedness problem we are able instead to obtain results for the whole existential rule language.

More on detail, the results we just presented are derived building on the following schema.

1. exhibit a property that a chase variant may enjoy, namely *Rank Preservation by Prime Ancestors*, and show that this implies the decidability of $\forall$-$k$-boundedness
2. show that the oblivious, semi-oblivious and restricted chases, as well as all their breadth-first versions, satisfy this property.

It is important to notice that proving point 1 gives us an even stronger result, that goes beyond Theorem 14. And indeed, this property has been used in [54] to prove that the problem is decidable also for other chases, including the parallel breadth-first restricted chase, the frugal chase [85] and one of its variants, called the vacuum-chase, proposed in [54].

The complexity of membership to $\mathsf{BN}[k]^X_\forall$ has been regarded in both [56] and [28]. Upper-bounds range from 2EXPTIME for $X \in \{\mathsf{obl}_{\mathsf{bf}}, \mathsf{sob}_{\mathsf{bf}}\}$ to 3EXPTIME for $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{res}, \mathsf{res}_{\mathsf{bf}}\}$. Membership is CO-NEXPTIME-complete for $\mathsf{Datalog} \cap \mathsf{BN}[k]^X_\forall$.

APPROACH On the technical side, the approach to prove the decidability of boundedness followed here can be called a *critical instance approach*. This is a useful scheme to decide properties related

| | obl | $\text{obl}_{\text{bf}}$ | sob | $\text{sob}_{\text{bf}}$ | res | $\text{res}_{\text{bf}}$ | eqv | $\text{eqv}_{\text{bf}}$ |
|---|---|---|---|---|---|---|---|---|
| Compliance Heredity by Factbase Restr. | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Rank Preservation by Prime Ancestors | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ |

Table 1.2: Chase variants with respect to properties

with the execution of the chase on all possible instances, and in particular its termination [96, 34, 88]. This approach will be recurrent also in the next sections, and in particular when studying linear rules. We allow ourselves to slightly generalize the prose of [34, 66] which we believe resumes the purpose of critical instances at best.

"*To decide a property $P$ on a rulebase it would be extremely useful to have a special factbase $F^*$ in place, let us call it critical, of a very simple form, that ensures the following: given a set of rules $R$, if there is a factbase that breaks the property $P$, then already $F^*$ does. With such a critical factbase in place, one can focus on the complement of our problem, and check if $P$ breaks on $F^*$*".

In [34, 66], where chase termination is studied, the property $P$ is the existence of an infinite (fair) derivation (i.e., a witness of non all-instance termination). Here, since we study $\forall$-$k$-boundedness, we will be looking for a $k+1$-deep derivation (i.e., a witness of non $\forall$-$k$-boundedness).

We now present a sufficient condition for a chase variant to enable a critical instance approach for deciding $\forall$-$k$-boundedness. As we look for derivations of a certain depth, this property must somehow concern the rank of atoms. As we look for a finite critical instance to run the chase, this property must somehow concern also the atoms of the input factbase.

**Definition 15** (Rank Preservation by Prime Ancestors)**.** *The X-chase* preserves rank by prime ancestors *if, for every X-derivation* $\mathsf{D}$ *from* $(F,R)$ *producing an atom* $\alpha$ *of prime ancestors* $A_\alpha \subseteq F$*, there is also an X-derivation* $\mathsf{D}'$ *from* $(A_\alpha, R)$ *producing* $\alpha$*, and moreover* $\mathsf{rank}_{\mathsf{D}}(\alpha) = \mathsf{rank}_{\mathsf{D}'}(\alpha)$.

The notion of *ancestors* of an atom in a derivation is quite standard. This follows like Definition 4. Let $\mathsf{D}$ be a derivation from $F$ and $R$ and $\mathsf{t} = \big((B,H),h\big)$ the *first* trigger of $\mathsf{D}$ producing the atom $\alpha$. Then the atoms in $h(B)$ are the *direct ancestors* of $\alpha$. Consider the transitive closure of this relation. Then, the ancestors of $\alpha$ that belongs to $F$ are called the *prime ancestors* of $\alpha$, and denoted by $A_\alpha$. Importantly, this set is finite and its size depends on the rank of the atom $\alpha$ itself, as well as on the maximum size (i.e., the number of atoms) of the body of a rule in $R$, we write $b$, hence $|A_\alpha| \leq b^{\mathsf{rank}(\alpha)}$.

**Proposition 16.** *If the X-chase preserves rank by prime ancestors, there is a critical instance* $\mathrm{cr}(R,k)$ *such that the following are equivalent.*

- $R \notin \mathsf{BN}[k]^X_\forall$
- *There exists an X-derivation from* $\mathrm{cr}(R,k)$ *of rank at least* $k+1$

In conclusion, if $X$ is a chase variant that preserves rank by prime ancestors, then $R$ belongs to $\mathsf{BN}[k]^X_\forall$ if and only if for every factbase $F$ of size at most $b^{k+1}$, every X-derivation from $(F,R)$ is of depth at most $k$. There is a finite number of non-isomorphic factbases of fixed size. Here, the critical instance (although arguably quite large) ranges over all such $F$. If the ruleset is not $k+1$ bounded on all instances then this holds already for one of these.

A second sufficient condition leading to the decidability of $\forall$-$k$-boundedness has also been presented. This is a stronger property, that actually implies rank preservation by prime ancestors.

**Definition 17** (Compliance Heredity by Factbase Restriction)**.** *The X-chase is* compliance hereditary by factbase restriction *if, for any X-derivation D from* $(F,R)$ *and subset* $A \subseteq F$*, the restriction of D obtained by retaining only the triggers that are applicable from the atoms in A is also an X-derivation.*

More from the technical side, let us mention that the need for the two properties we just presented is justified by the fact that when restricting a derivation to the triggers that are applicable on a subset of its initial atoms two phenomenons can occur [56]. The first is that the rank of atoms may increase (hindering the test of a particular bound $k$). The second is that the compliance condition may be broken (hindering the test of a particular variant). Table 1.2 resumes the results obtained concerning the two presented properties. Note that the breadth-first versions of the semi-oblivious and restricted chase do not enjoy compliance heredity. Indeed, by restricting a $X_{\mathsf{bf}}$-derivation we may not have a $X_{\mathsf{bf}}$-derivation anymore, as there may be triggers that were not applicable that become applicable by starting from the prime ancestors of an atom. This is illustrated by Example 26 and 27 of [56].

Finally, Example 18 of [56] shows that the (breadth-first) equivalent-chase does not enjoy any of the above mentioned property. The decidability of the membership to $\mathsf{BN}[k]^{\mathsf{eqv}}_\forall$ is still open. For the case of the equivalent chase, it remains to be shown whether a critical instance exists to apply again the resolution scheme we developed. The decidability of the existential variant of the problem $\exists$-$k$-boundedness is also open for many cases. Nevertheless, the results on $\forall$-$k$-boundedness also imply the decidability of the existential version of the problem for some chase variant. Membership to $\mathsf{BN}[k]^X_\exists$ is decidable when $\mathrm{X} \in \{\mathsf{obl}, \mathsf{obl}_{\mathsf{bf}}, \mathsf{sob}, \mathsf{sob}_{\mathsf{bf}}\}$. It follows from Proposition 19 of [56] that $\mathsf{BN}[k]^X_\exists = \mathsf{BN}[k]^{X_{\mathsf{bf}}}_\exists = \mathsf{BN}[k]^{X_{\mathsf{bf}}}_\forall$ when $X \in \{\mathsf{obl}, \mathsf{sob}\}$.

### The Case of Linear Rules

We now turn our attention to the boundedness property (where the bound $k$ is not part of the input) and focus on an important fragment of existential rules called *linear*. It is worth mentioning that the investigations carried for linear rules concerned the problem of chase termination, and the results will be presented under this prism. Nevertheless, chase termination matters a lot for boundedness because the two notions actually coincide for linear existential rules. A rule is called linear if its body

is reduced to a single atom. These rules form a simple yet important subclass of rules which generalizes inclusion dependencies [58] and positive inclusions in DL-Lite$_R$ (the formal basis of the web ontological language OWL 2 QL) [40]. Linear rules are known to be FO-rewritable [36], therefore, answers to queries can always be computed by query rewriting when reasoning within this fragment of existential rules. Nevertheless, it is still interesting to study the chase for linear rules because in many cases the worst-case-exponential size and the unusual form of the rewritten query may rise practical efficiency issues. Hence, the materialization inferences in the data is often a good alternative to query rewriting, provided that some chase algorithm terminates.

CONTRIBUTION At a glance, the contribution of our work is twofold. From the *decidability* point of view, (1) it provides alternative proofs and novel positive results for chase termination (equivalently boundedness) on *single-head* linear rules (that is, linear rules where the head is reduced to a single atom). From the *algorithmic* point of view, (2) it presents a unified approach to decide termination exploiting a simple data-structure called *derivation trees* [17, 118] which gives a constructive approach which is more likely to lead to an effective implementation.

Before detailing our results, let us contextualize it with respect to related work. Some positive results had already been obtained for linear rules, and actually for a superset of them, namely *guarded rules*.[6] Decidability has been shown for the oblivious and semi-oblivious chase [33] as well as for the core chase termination on a fixed instance [74]. In passing, let us also mention that, independently from our work, the decidability for the all-instance termination for the restricted chase on single-head guarded rules has been shown in a recent paper [66] reducing the problem to satisfiability of monadic second order logic.

Our approach sharply distinguishes by being constructive: we provide an algorithm that decides termination and, given a fixed instance, can actually even compute the result of the chase. In case of non-termination our algorithm precisely pinpoints a forbidden pattern leading to non-termination. Summing up, our results are the following.

- a new proof of the decidability of semi-oblivious chase termination, using different objects than [33]; our algorithm has the same complexity upper-bound;
- the decidability of the restricted chase termination, for both versions of the problem, i.e., termination of all (fair) chase sequences and termination of some (fair) chase sequence;
- a new proof of the decidability of the core chase termination, with different objects than previous work on the core chase termination reported in [74].

APPROACH Our resolution scheme relies again on critical instances for deciding chase termination. Critical instances for deciding the termination of linear rules (even multi-head ones) can be made out of atomic instances, i.e., factbases containing a single atom [36, 33, 88]. For linear rules, the oblivious, semi-oblivious, restricted, and equivalent chase terminates on all instances if and only if it terminates on all atomic instances. This allows us also to see that chase termination and bounded-

[6] guarded rules are rules where the body has an atom containing all body variables, and linear rules are trivially guarded.

ness coincide for linear rules. Indeed, if the $X$-chase terminates on an atomic instance $\alpha$ then it does so with a derivation of a certain depth $k_\alpha$. As there are a finite number of (isomorphic) atomic instances it is possible to identify an (uniform) upper bound which is the maximum of the depths of all possible derivations from atomic instances. Hence, chase termination and boundedness coincide. This has been also shown in [54] and [28].

**Proposition 18** ([88]). *For a set $R$ of single-head linear rules and a chase variant $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{res}, \mathsf{eqv}\}$ there is a critical instance* $\mathrm{cr}(R)$ *such that the following are equivalent.*

- $R \notin \mathsf{CT}^X_\forall$
- *There exists an infinite $X$-derivation from* $\mathrm{cr}(R)$

Note that this statement does not talk about *fairness*, which greatly simplifies the problem study. Fairness can be neglected here because of a very important property stating that the existence of an infinite fair derivation implies the existence of an infinite derivation (and, to stress that again, possibly not fair) [33, 88]. When moving to multi-head linear rules, things are different. While the property still holds for the oblivious and semi-oblivious chase, this is not the case for the restricted and core chase. That is, there may be infinite restricted chase derivations, none of which is fair, and this holds even for arity 2 predicates [88]. So, both in [88] and [66] the restriction on single-head rules is really key for deciding chase termination. The case of multi-head rules is a more elusive problem than it could seem at a first glance. For concision, in Proposition 18 we focussed on the $\forall$ version of chase termination. We already recalled that $\mathsf{CT}^X_\forall = \mathsf{CT}^X_\exists$ when $X \in \{\mathsf{obl}, \mathsf{sob}, \mathsf{eqv}\}$. This does not hold for the restricted chase, but the approach extends to decide membership for $\mathsf{CT}^X_\exists$ [88].

With this characterization in hand, the next step is to search for infinite derivations. Towards this goal, we exploit in a novel way a tree structure, called *derivation tree*, which was originally introduced in [17, 118] to answer queries over greedy-bounded treewidth sets of existential rules. In the remainder of the section we focus on presenting the use of derivation trees to show the decidability of the all sequence termination problem for the semi-oblivious and restricted chase variant. We will then conclude by discussing the extension of the approach to the other variants of the problem. Roughly speaking our approach is to build derivation trees from critical instances and then check for the presence of forbidden patterns within them, called *unboundedness witness* which can lead to infinite derivations. Let us start by defining derivation trees.

**Definition 19** ([17, 118]). *Let $\mathsf{D} = (r_1, h_1), (r_2, h_2), \ldots$ be a derivation from $F$ and $R$ producing a sequence of atoms $\alpha_0, \alpha_1, \ldots$ such that $F = \{\alpha_0\}$ and $\alpha_i = h_i^{safe}(\mathrm{head}(r_i))$ for all $i > 0$.*

*The derivation tree assigned to $\mathsf{D}$ is a tree defined as follows.*

- *the root of the tree is $\alpha_0$*
- *the atom $\alpha_j$ is the child of $\alpha_i$, with $i < j$ the smallest integer such that $h_j(\mathrm{fr}(r_j)) \subseteq \mathrm{terms}(\alpha_i)$*

The crux of the definition is that an atom which is generated by a trigger $(r_j, h_j)$ is attached to the highest atom in the tree containing $h_j(\mathrm{fr}(r_j))$, that is, the image of the rule frontier. Accordingly, the terms in $h_j(\mathrm{fr}(r_j))$ are also called the *shared terms* of $\alpha_j$. Indeed, these are the terms that an atom $\alpha_j$

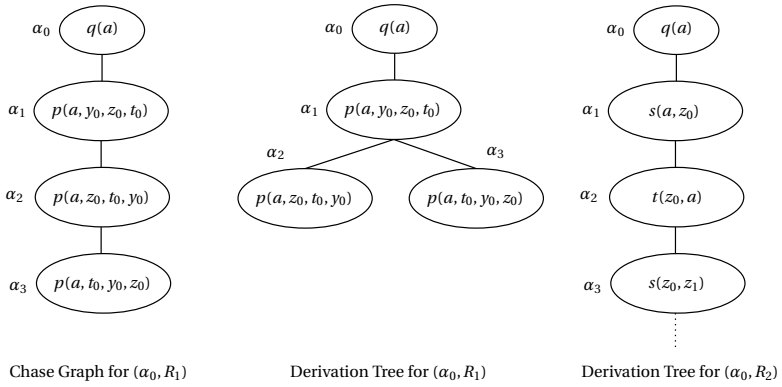



Figure 1.1: Chase Graph and Derivation Trees of Examples 20 and 22

is sharing with its parent $\alpha_i$. Note that $\alpha_0$ has no shared terms. Finally, the *positions* associated with the shared terms of an atom are called *shared positions*.

To illustrate derivation trees and their interest, we put is side by side with a more common structure, which is the *chase graph* associated to a derivation. This is build similarly as a derivation tree. The root of the chase graph is an atom $\alpha_0$. But, the atom $\alpha_j$ is the child of $\alpha_i$ if $h_j(\text{body}(r_j)) = \alpha_i$, that is if $\alpha_i$ is the atom where the rule $r_j$ generating $\alpha_j$ has been applied. Note also that for linear rules, the chase graph trivially reduces to a tree.

**Example 20.** *Consider the rules* $r = q(x) \rightarrow \exists y \exists z \exists t\ p(x, y, z, t)$ *and* $r_{perm} = p(x, y, z, t) \rightarrow p(x, z, t, y)$ *and the knowledge base* $(\{\alpha_0\}, R_1)$ *where* $R_1 = \{r, r_{perm}\}$. *Let* $\mathsf{D}$ *be the derivation where rules* $r$ *and* $r_{perm}$ *are applied generating the atoms* $\alpha_1 = p(a, y_0, z_0, t_0)$, $\alpha_2 = p(a, z_0, t_0, y_0)$, $\alpha_3 = p(a, t_0, y_0, x_0)$. *The chase graph and derivation tree for the derivation are pictured in Figure 1.1.*

In a nutshell, compared with the chase graph, the derivation tree makes more explicit the relationships between two inferred atoms in terms of the freshly introduced existential variables. For example, atoms $\alpha_2$ and $\alpha_3$ in the derivation tree have the same parent $\alpha_1$, which is the node closest to the root containing all of their shared terms. The idea we will exploit is that if there is a "similar" atom that can be inferred twice along a path of the derivation tree then this will go on forever in an infinite derivation. To detect this kind of pattern however, a basic chase graph will not suffice. As illustrated in Figure 1.1, $\alpha_1, \alpha_2, \alpha_3$ are "similar", but this derivation is finite. It is perhaps time to precise the notion of "similar" atoms. This is done by the following definition of *unboundedness witness*.

**Definition 21** ([88])**.** *An* unboundedness witness *(UW) in a derivation tree is a pair of distinct nodes* $(\alpha, \alpha')$ *such that:*

- $\alpha$ *is an ancestor of* $\alpha'$
- $\alpha$ *and* $\alpha'$ *have the same type*
- $\alpha$ *and* $\alpha'$ *have the same shared positions*

The notion of type of an atom $\alpha$ is standard. This is a pair $(p, P)$ where $p$ is the predicate of the atom and $P$ is a partitioning of it positions such that two positions sharing the same terms are in the same

partition.[7] Going back to Figure 1.1, note that $\alpha_1$ and $\alpha_3$ have the same type but different shared positions in the derivation. Hence, this couple does not form an UW. Also, as already said, note that $\alpha_2$ and $\alpha_3$ in the chase graph do form an UW (provided that the notion of shared positions is extended to the chase graph) but, the derivation is not infinite and this well illustrates that the structure not adequate to decide chase termination because it does not capture well the relationships between freshly generated nulls. Let us conclude with an example of UW in a derivation tree capturing the existence of an infinite derivation.

**Example 22.** *Consider the the knowledge base $(\alpha_0, R_2)$ where the ruleset $R_2$ contains the following rules $r_{q\,next_s} = q(x) \to \exists z\ s(x,z)$, $r_{s\,inverse_t} = s(x,y) \to t(y,x)$, and $r_{t\,next_s} = t(x,y) \to \exists z\ s(x,z)$. Consider the derivation built from the application of the rules in the given order. The derivation tree is pictured in Figure 22. In this case, the atoms $\alpha_1$ and $\alpha_3$ form an UW. And indeed, the derivation can be extended to an infinite (fair) derivation.*

**Proposition 23** ([88])**.** *For $X \in \{\mathsf{sob}, \mathsf{res}\}$, there exists an infinite derivation from $\alpha$ and $R$ if and only if there exists an $X$-derivation tree associated with $\alpha$ and $R$ that contains an unboundedness witness.*

This gives us a constructive algorithm for verifying chase termination. Starting from every critical instance, we compute all possible derivation trees. Since there are a finite number of types and shared positions this halts either naturally or a the first encountered unboundedness witness.

The approach can be extended to decide membership to $\mathsf{CT}^{\mathsf{res}}_{\exists}$, which is equivalent to having a finite derivation from all critical instances. Hence, for any canonical instance, we build all restricted derivations until either *(i)* there is an unboundedness witness in their associated derivation tree or *(ii)* there is no active trigger anymore. The chase terminates if and only if for all instances, there is a least one restricted derivation that halts because of the second condition [88].

For the core chase termination, the notion of derivation tree is extended to that of entailment tree. The main difference with a derivation tree is that it employs a more general parent-child relationship, that relies on entailment rather than on rule application, hence the name entailment tree. We refer to [88] for a detailed presentation. A rough analysis of our approach also gives us some upper-bounds on the complexity of checking boundedness.

**Proposition 24** ([88])**.** *For single-head linear rules, membership is decidable for $\mathsf{CT}^{X}_{\star}$, with $\star \in \{\forall, \exists\}$ and $X \in \{\mathsf{sob}, \mathsf{res}, \mathsf{eqv}\}$. It is in* PSPACE *for $\mathsf{CT}^{\mathsf{sob}}_{\forall}$, in* CO-N2EXPTIME *for $\mathsf{CT}^{\mathsf{res}}_{\forall}$, in* N2EXPTIME *for $\mathsf{CT}^{\mathsf{res}}_{\exists}$, and in* 2EXPTIME *for $\mathsf{CT}^{\mathsf{eqv}}_{\forall}$.*

Future work here concerns the study of the precise complexity of the termination problems, except for the case of the semi-oblivious chase for which our algorithm can be make run in polynomial space (which is optimal [33]). For the other cases, we believe that a finer analysis can provide much tighter bounds. We already mentioned the difficulties raised by complex-head rules and fairness. It is an open question whether our approach can be extended to more complex classes of existential rules from the guarded family, for which derivation trees have been introduced.

[7] For example, the type of $p(a,x,y,x)$ is $(p, [\{1\}, \{2,4\}, \{3\}])$

We now turn to the third question that has been investigated: a characterization of boundedness in terms of known properties of rulesets. Actually, this was even one of the first issues that had been regarded [89] when first approaching the study of boundedness. What are the precise relationships between boundedness, chase termination and FO-rewritability? We already mentioned that boundedness entails chase termination and FO-rewritability. Seen this, an intriguing question is whether boundedness is *exactly* the intersection of chase termination and FO-rewritability. Why should this be the case? The basic intuition is the following. On one side, chase termination ensures that the knowledge base has a finite universal model. Hence, the chase does not need to generate an infinite number of fresh existential variables to compute its output. On the other side, FO-rewritability makes that query rewriting process will not need to generate an infinite set of rewritings. By the equivalence of the forward and backward chaining [15], this is a sign of the fact that the existence of an atom in the chase cannot depend (in a certain sense) on an arbitrary number of other atoms. Also suggesting this is that $\mathsf{BN} = \mathsf{FO\text{-}R}$ for datalog rules [12]. But is this the correct vision of the situation?

Seen the number of boundedness classes we defined, we need first to precise the questions that we are aim at. At first, Example 11 warns us that we have to be careful when $\forall$ boundedness is considered. Indeed, the example shows a ruleset which is both in $\mathsf{CT}^X_\forall$ and $\mathsf{FO\text{-}R}$ but does not belong to $\mathsf{BN}^X_\forall$. We will come back to this later. So, our interest here is on characterizing the existential classes bounded rules, for which we pose the following conjecture.

**Conjecture 25.** $\mathsf{BN}^X_\exists = \mathsf{CT}^X_\exists \cap \mathsf{FO\text{-}R}$

CONTRIBUTION Briefly, our main result shows that the conjecture holds for the oblivious and the semi-oblivious chase.

**Proposition 26** ([28])**.** *It holds that*

- $\mathsf{BN}^{\mathsf{obl}}_\exists = \mathsf{CT}^{\mathsf{obl}}_\exists \cap \mathsf{FO\text{-}R}^{\mathsf{AF}}$
- $\mathsf{BN}^{\mathsf{sob}}_\exists = \mathsf{CT}^{\mathsf{sob}}_\exists \cap \mathsf{FO\text{-}R}$

It is important to recall that almost all known sufficient conditions for chase termination fall within these chase variants, rich-acyclicity [75], weak-acyclicity [60] and acyclic-GRD [15], MFA [71], at the exception of [44] which applies to the restricted chase variant. Hence, these could be exploited together with FO-rewritability to concretely verify membership to $\mathsf{BN}^{\mathsf{obl}}_\exists$ and $\mathsf{BN}^{\mathsf{sob}}_\exists$.

These results reveal several nuances between the oblivious and the semi-oblivious case. Before presenting them, we need to introduce two notions. The first is the subclass of conjunctive queries called, *full atomic* queries, noted as $\mathsf{AF}$. These are queries with a single atom and only answer variables. The second is that of *forced existential* rule: these are rules where *all head atoms* are forced to have at least one existential variable.[8] Note that every ruleset can be decomposed in a set made

[8] these are called fully-existential rules in [28], but they should not be confused with that of full-TGDs, that are range restricted (or datalog) existential rules

only of datalog and forced existential rules and this preserves the oblivious and semi-oblivious chase termination [28].

The case of the oblivious chase is quite peculiar, due to its weaknesses at detecting redundancies. Let us focus on the particular case of forced existential rules. We have that chase termination even implies boundedness, and so $\mathsf{CT}^{\mathsf{obl}}_{\exists} = \mathsf{BN}^{\mathsf{obl}}_{\exists}$. This gives a somewhat twisted result that $\mathsf{CT}^{\mathsf{obl}}_{\exists} \subset \mathsf{FO\text{-}R}$, that is, chase termination even leads to FO-rewritability. This does not hold for the semi-oblivious chase, where forced existential rules do not behave differently from general existential rules.

A more significative difference is that, assuming $R \in \mathsf{CT}^{\mathsf{obl}}_{\exists}$, we have $\mathsf{FO\text{-}R} = \mathsf{FO\text{-}R}^{\mathsf{AF}}$, that is the notion of FO-Rewritability collapses to a much stronger notion of rewritability $\mathsf{FO\text{-}R}^{\mathsf{AF}} \subset \mathsf{FO\text{-}R}$, which asks that only full atomic queries are FO-rewritable. This again is intimately related with the vision of a ruleset given by the datalog/forced-existential decomposition and by the way in which the oblivious chase generates fresh terms. Again, none of this hold for the semi-oblivious chase.

APPROACH Again for the sake of presentation, to illustrate our approach we will focus on a single chase variant: the semi-oblivious chase. We refer to [28] for details on the oblivious chase. To compute a bound on the rank of atoms produced by the chase we build on a notion of *existential distance*[9] for the fresh nulls inferred by the chase. Intuitively, this should account for the strata of existential variables that have to be generated to produce a certain term.[10] This notion is similar yet different from that of rank (Definition 4) which accounts for the strata of rule applications, some of which may generate a new atom but *no new terms*. As a last point of detail, we stress again that this notion of existential distance is different for the oblivious and semi-oblivious chase. The notion of *frontier existential distance* we give below is suitable to study all chases that are at least as expressive as the semi-oblivious chase.

**Definition 27.** *The* frontier *existential distance of a term $t$ in a derivation $\mathsf{D}$ from $F$ and $R$ is*

$$\mathsf{distance}^{\mathsf{fr}}_{\exists}(t) = \begin{cases} 0 & \text{if } t \in \mathrm{terms}(F) \\ 1 + \max\{\, \mathsf{distance}^{\mathsf{fr}}_{\exists}(v) \,\} & \text{otherwise} \end{cases}$$

*assuming that $(r, h)$ is the* first *trigger of $\mathsf{D}$ producing $t$ and $v$ is any term in $h(\mathrm{fr}(r))$.*

*The frontier existential distance of a derivation is the maximum frontier existential distance of its terms if the derivation is finite, and infinite otherwise.*

We then build again on critical instances. It is known that both the oblivious and the semi-oblivious chase terminate if and only if they terminate on the critical instance. It can be shown that the existential distance of a derivation from the critical instance upper bounds that of any derivation. Then, FO-rewritability ensures that once a set of terms appear in the chase at rank, there will be a bound for the rank at which all atoms using those terms as frontier variables appear.

[9] this is called existential depth in [28]. Here we rename it so as to not confuse it with the depth of derivations

[10] this can be assimilated with the number of levels in a derivation tree, although this notion only applies to greedy bounded treewidth sets of rules [118]

**Proposition 28** ([28]). *When* $R \in \mathsf{CT}^{\mathsf{sob}}_{\exists}$ *there exists a constant* $k$ *such that for all* sob*-derivation* D *using the rules of* $R$ *it holds* $\mathsf{distance}^{\mathsf{fr}}_{\exists}(\mathsf{D}) \leq k$.

*When* $R \in \cap\mathsf{FO\text{-}R}$, *there exists a constant* $k$ *such that for all* sob*-derivation* D *using the rules of* $R$ *it holds* $\mathsf{rank}(\mathsf{D}) \leq \mathsf{distance}^{\mathsf{fr}}_{\exists}(\mathsf{D}) \times k$.

Overall the characterization of boundedness in terms of chase termination and FO-rewritability allows us to better understand the property, and delve on the interaction between chase termination and FO-rewritability. Concerning decidability and complexity, we already mentioned that membership $\mathsf{BN}^{X}_{\forall}$, $\mathsf{CT}^{X}_{\forall}$ and FO-R is undecidable for general existential rules. Importantly, new decidability and complexity results about boundedness for specific existential rules studied in the literature can be obtained as direct corollaries of our results. This is in particular the case for classes known to be FO-rewritable.

**Corollary 29.** *([28]) For any class of existential rules* $\mathscr{C} \in \mathsf{FO\text{-}R}$ *and chase variant* $X \in \{\mathsf{obl}, \mathsf{sob}\}$ *it holds that* $\mathscr{C} \in \mathsf{BN}^{X}_{\exists}$ *iff* $\mathscr{C} \in \mathsf{CT}^{X}_{\exists}$.

This implies that membership to $\mathsf{BN}^{\mathsf{obl}}_{\exists}$ and $\mathsf{BN}^{\mathsf{sob}}_{\exists}$ is PSPACE-complete for the two main classes of FO-rewritable existential rules, namely *linear* and *sticky*. This matches also the results of [88]. Indeed, deciding $\mathsf{CT}^{\mathsf{obl}}_{\exists}$ and $\mathsf{CT}^{\mathsf{sob}}_{\exists}$ is PSPACE-complete for both [33, 88, 34]. We also get an upper bound on the complexity of membership to $\mathsf{BN}^{\mathsf{obl}}_{\exists}$ and $\mathsf{BN}^{\mathsf{sob}}_{\exists}$ for a major class of existential rules, namely *guarded*. This class is neither $\mathsf{CT}^{\mathsf{sob}}_{\exists}$ nor FO-R. However, membership to $\mathsf{CT}^{\mathsf{obl}}_{\exists}$ and $\mathsf{CT}^{\mathsf{sob}}_{\exists}$ for guarded rules is decidable in 2EXPTIME [33]. Then, a careful reduction from [18] allows us to set the result. The paper shows that checking FO-rewritability for a single query under guarded rules is in 2EXPTIME. This suffices since to show Proposition 28 one just needs to consider FO-R on a polynomial number of queries [28].

Our quest for the characterization of boundedness showed that Conjecture 25 holds for the oblivious and semi-oblivious chase. For the core chase, the conjecture $\mathsf{BN}^{\mathsf{core}}_{\exists} = \mathsf{CT}^{\mathsf{core}}_{\exists} \cap \mathsf{FO\text{-}R}$ or, equivalently, $\mathsf{BN} = \mathsf{CT} \cap \mathsf{FO\text{-}R}$ has been formulated in [89]. The reported proof is however incorrect. This conjecture has been successively attacked by a recent work and proved true for a subset of existential rules that form *local-theories* [102]. The conjecture for the core chase reveals to be elusive, and is still open for general existential rules. The problem has not been regarded so far for the restricted chase.

Overall, our results suggest the following vision of the boundedness property. To explain that let us introduce novel class of rules called "limited (existential) distance" LD inspired by the notion of frontier existential distance we just defined. Again, below we consider chase variants that are at least as expressive as the semi-oblivious chase.

**Definition 30.** $R \in \mathsf{LD}^{X}_{\exists}$ *if there exists* $k$ *such that, for every factbase* $F$*, there exists a fair* $X$*-derivation with* $R$ *such that* $\mathsf{distance}^{\mathsf{fr}}_{\exists}(\mathsf{D}) \leq k$.

What can be directly deduced from Proposition 28 is that $\mathsf{BN}^{X}_{\exists} = \mathsf{LD}^{X}_{\exists} \cap \mathsf{FO\text{-}R}$ for every chase variants which is at least as expressive as the semi-oblivious chase. Our work showed that $\mathsf{CT}^{\mathsf{sob}}_{\exists} = \mathsf{LD}^{\mathsf{sob}}_{\exists}$.

This again is related by the expressivity of the semi-oblivious chase [87]. We believe however that this relationship is unlikely to hold for the restricted and core chase. For these variants the question is whether the notions of $\mathsf{CT}^X_\exists$ and $\mathsf{LD}^X_\exists$ collapse when FO-R holds, that is $\mathsf{CT}^X_\exists \cap \text{FO-R} = \mathsf{LD}^X_\exists \cap \text{FO-R}$.

Finally, a characterization of the forall version of boundedness is still open. Example 11 shows that $\mathsf{BN}^X_\forall \neq \mathsf{CT}^X_\forall \cap \text{FO-R}$ for all chase variant $X$. The reason should be found within the notion of FO-R, which is too powerful. In Definition 10, we introduced a stronger property that we believe is promising for solving the case. Our conjecture is the following.

**Conjecture 31.** $\mathsf{BN}^X_\forall = \mathsf{CT}^X_\forall \cap \mathsf{RT}$

OUTLINE Overall, the study of boundedness led us to a better understanding of the chase. We have seen the importance of breadth-first derivations for the oblivious and semi-oblivious chase, to be bounded uniformly over all instances. We have seen some important connections between chase termination and FO-rewritability, whose intersection is exactly boundedness in some cases. Finally boundedness led us to reconsider also our understanding of the first-order rewritabilty, which may still deserve some investigations, despite the fact that it already concerns a relevant portion of the literature on ontology-mediated query answering. The open questions we presented will be subject of our future work.

## 1.4 Reasoning on NoSQL Databases

This research track stem from the will of extending the ontology-mediated query answering approach towards JSON data and document oriented key-value stores. Simply put, seen that inferences permit to get more out of data by handling incompleteness and intensionality, why should this be limited only to relational and RDF databases? Two reasons in particular make this question relevant. The first is the widespread diffusion of the JSON format. In the last years, JSON became one of the principal standards for exchanging data, surpassing in many contexts XML (although this last one still remains the reference in many applications and Web standards). Massive collections of JSON datasets are produced, and this does not concern only the Web, and there is an interest at exploiting them at best. The second reason is the ascent of a breed of database systems, called NoSQL systems. NoSQL databases such as document oriented key-value stores are tailored for this JSON data, and recognized for their performances. Implementations include systems as MongoDB [4] and CouchDB [1]. However, also well established solutions like PostgreSQL [6], Oracle NoSQL Database [5], IBM Informix [3] today provide support for JSON data.

So, not only we have massive datasets to exploit, but also efficient systems that can be used as a foundations to perform reasoning. Indeed, a very important factor for the success of ontology-based data access has been the possibility of deploying reasoners on top of existing relational and RDF databases. Disposing of mature database technology allows a reasoner to delegate many intermediate query evaluation steps. Query evaluation boils down to computing a physical plan for accessing data, whose finding is a non-trivial task that has been matter of databases research for decades.

However, while the use or relational and RDF databases has been a lot investigated for querying data with ontologies, the extension of the paradigm to NoSQL databases has been little regarded. Handling heterogeneous data like JSON has been done typically considering a more data-integration oriented approach, where data is mapped to an ontological schema through JSON-to-RDF mappings [27]. The approach that has been pursued in our case was instead to add a set of rules directly on top of the data, with the aim of better explaining and exploiting the data itself. Therefore, our main goal here is to *design decidable rule languages* for reasoning on JSON data stored on NoSQL databases.

The setting we consider asks to cope with certain constraints. The first of course is to deal with database APIs that accept only tree-like data and queries. The second is to privilege virtualization and query rewriting based approaches. Indeed, not only the systems we target provide less rich update facilities, and lower performances with respect to queries, but also the fact that data must have a tree-like shape may quickly lead to an unpractical data blowup if saturation is preferred. In short, this means that we are seeking for decidable rule languages that are also rewritable into native queries for the NoSQL system, and that furthermore admit sound and complete sets of tree-like rewritings.

Let us mention that the framework presented here can be compared to existing languages designed for reasoning on nested structures. For instance, Frame Logic [82, 81] provides a logical foundation for frame-based and object-oriented languages for data and knowledge representation. The Elog rule language [22] underlying the the Lixto system [21] is a fragment of monadic Datalog [69] that

```
{ dept:{
    name:"Computer Science"
    professor:[
        {name:"Alice"  reachable:"yes"  boss:"Charles"}
        {name:"Bob"  phone:{office:"5-256"}  } ]
    director:null  } }
```

Figure 1.2: Key-value record example

has been specifically designed for extracting tree shaped data from HTML pages. Active XML [9] is a formalism to model distributed systems represented as trees with function calls for tasks such as sending, receiving and querying data (like web services). In all of these cases a forward-chaining approach for reasoning was applied. Our approach is sharply different, in that we seek at *i*) developing reformulation based techniques for query answering and *ii*) reusing NoSQL database technology.

THE JOURNEY I started working on reasoning on JSON data after arriving in Montpellier. The first investigations have been done with Marie-Christine Rousset and Marie-Laure Mugnier. In this collaboration we studied rules and queries based on paths, and focussed in particular on decidable languages obtaining with atomic rules where either the body or the head of rules is of size one. This work continued in a collaboration with Pierre Bourhis, Meghyn Bienvenu, Marie-Laure Mugnier and Sophie Tison. In this context a new language based on contextual suffix rules has been proposed, and an important connection between query-rewriting (in an OMQA setting) and word-rewriting (as intended in tree rewriting systems) leading to decidability has been shown. This line of work continued in the context of the thesis of Olivier Rodriguez that I co-supervise with Marie-Laure Mugnier, where the target is to reason with more expressive tree-like queries and rules.

### QUERYING JSON DATA WITH RULES

To illustrate our approach, consider the JSON record in Figure 1.2 (commas are omitted). It contains several different keys among which four (`name`, `reachable`, `boss`, `office`) are associated with constant values. The key `professor` is associated with a sequence of two records, one for each professor, and the key `director` with an unspecified value *null*. Key-value stores query language typically build on a set of low-level operations to query and update keys and values, like get(k), put(k, v), and clear(k). These systems are designed and optimized for this kind of operations that can be performed in parallel over large collections of records. Since we are interested in *querying* KV stores with rules, we will focus on get() queries. An example of query retrieving names of professors is get(`dept.professor.name`). This expression returns the values "*Alice*" and "*Bob*" once evaluated on the record of Figure 1.2. Another feature supported by KV stores is the possibility to check structural properties of a record before selecting some content. Thus, we further look at queries with a check() construct like check(`dept.director`). These condition can also be combined to make for instance that the get part of the query is evaluated at the only condition that the check part returns true. The

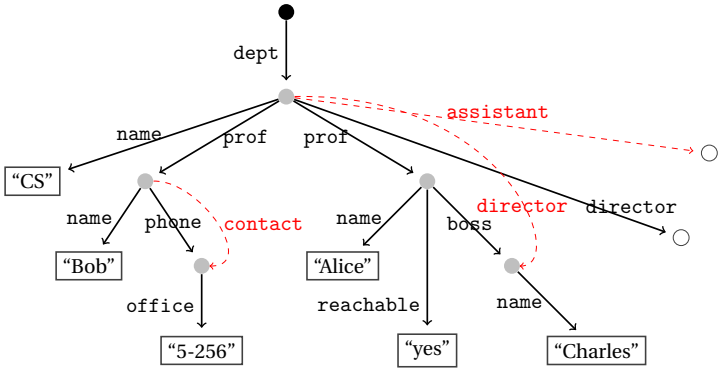



Figure 1.3: Tree associated to the record of Figure 1.2 and rule application

check() construct is particularly useful when dealing with incomplete information (values for which we just know the existence) which is expressed by null values, like for the key `director` in Figure 1.2.

To illustrate the role of rules, consider the query get(`dept.professor.contact.office`), searching through the contacts of all professors. Although in the record of Figure 1.2 there is a phone record this query yields no result, simply because the key `contact` is not found. Here is where rules comes into play. In fact, we could simply introduce some general knowledge and equip the store with a rule saying that "*any value for the key* `phone` *is also a value for the key* `contact`":

$$\sigma_1 : \texttt{phone} \longrightarrow \texttt{contact}$$

thereby retrieving the value "5-256" associated to the key `office` in the subrecord. Furthermore, rules able to assert the existence of a value can be used to reason with incomplete information. For example, the following rule says that whenever we find a value for the key `director` in a record, there exists a value for the key `assistant` (although this could be unspecified):

$$\sigma_2 : \texttt{director} \longrightarrow \exists \texttt{assistant}$$

With this knowledge in hand, the evaluation of the query `check(dept.assistant)` on the record of Figure 1.2 yields true, although there is no explicit value for the key `assistant`.

These examples illustrate the general approach we are following and how adding rules can help exploiting JSON data. Our main question now is to define query and rule language that can be used to reason on top of JSON data. As we already mentioned, we moreover want languages that allow us to rewrite queries into the key-value store native query languages.

### Reasoning Framework

We now present our setting for reasoning on key-value stores. A JSON record, or *key-value record*, is a finite set of key-value pairs. A *value* is recursively defined as (*i*) a constant or a null, (*ii*) a sequence of values $[e_1 \dots e_n]$, or (*iii*) a record of the form $\{(\mathtt{k}_1, e_1) \dots (\mathtt{k}_n, e_n)\}$ where each $\mathtt{k}_i$ is a distinct label

(among $\mathtt{k}_1,\ldots,\mathtt{k}_n$) called *key* and each $e_i$ is a value. We assume that both sequences and records are always non-empty. A KV-store is a set of key-value records.

JSON records can be abstracted by unranked, unordered, and rooted, trees where all edges are labelled by keys, and leaf-nodes are either labelled by constants or unlabelled (if originally a *null* value was employed). All non-leaf nodes of the tree are unlabelled. A key-value pair $(\mathtt{k}, e)$ where $e$ is a sequence is represented by several edges labeled by $\mathtt{k}$ leading to the nodes that represent the elements of $e$.[11] From now on, we will simply refer to these objects as *trees*. We call *paths* the trees which form a path. Then, to account for reasoning later, it is helpful to further extend these structures from trees to directed acyclic graphs (DAGs). Hence, below we will denote by $J$ a DAG. Moreover we will write root($J$) for the root of $J$ and leaves($J$) for the set of leaves of $J$, that is the nodes which have no children. Figure 1.3 gives the tree representation of the record in Figure 1.2.

The principal query facilities offered by key-value stores concern the evaluation of tree-like queries, and path queries are the staple of the query language [4, 1]. Hence, path queries and rules will be the starting point of our study, as these are the simplest type of objects one can conceive on JSON records. We will see however that, despite the apparent simplicity, they provide a great (and even too large) expressivity.

In the definitions that will be following, we will recurrently use the notation $\mathscr{L}$ to denote a set of nodes of the query (or the rule) that has to be mapped to some labelled leaves of $J$. For path queries $\mathscr{L}$ trivializes to either the empty set or the set containing the leaf of $P$. As we will see, this facility will become key for designing new rule languages with decidable query answering.

**Definition 32.** *A* path *query is a pair $q = (P, \mathscr{L})$ where:*

- *$P$ is a path*
- *$\mathscr{L} \subseteq$ leaves($P$)*

*A tuple of constants* $\mathbf{a}$ *is an answer to a query $q$ over $J$ if there is a rooted homomorphism $h : P \to J$. Furthermore, it must be that either $\mathscr{L} = \emptyset$ and $\mathbf{a} = ()$ or $\mathscr{L} = \{n\}$, $h(n)$ is labelled by $v$, and $\mathbf{a} = (v)$.*

*The set of answers to $q$ on $J$ is denoted by* ans($q, J$).

It is easy to see that check queries like `check(dept.director)` can be seen as the queries such that $\mathscr{L} = \emptyset$ and get queries like `get(dept.professor.name)` are otherwise. Finally, to be compliant with NoSQL systems, for query answering we consider *rooted* homomorphisms, that are simply homomorphisms mapping root($P$) to root($J$).

Let us now define path rules. In the examples, we shall use the notation `Body` $\to$ `Head` to describe a rule (or `Context : Body` $\to$ `Head` later when adding contexts), but for the sake of formal presentation we model them as two paths sharing the same root and (possibly) the same leaf.

**Definition 33.** *A path rule is a pair $\sigma = (B, H)$ where $B$ and $H$ are two paths sharing the same root.*

*When $B$ and $H$ also share the same leaf node, the rule is called a* path inclusion.

*When $B$ and $H$ share only the root, the rule is called a* mandatory path.

[11] We do not represent the ordering on the elements of a sequence. Note that a position can always be simulated by a key.

Again, no internal node (different from the root or the leaf) can be shared by the paths. We also consider that $B$ and $H$ contain at least one edge. As usual we will write body($\sigma$) for $B$ and head($\sigma$) for $H$. The rule $\sigma_1$ presented before illustrates a path-inclusion, while the rule $\sigma_2$ is a mandatory path.

The notion of rule application follows. A rule $(B,H)$ is said to be applicable on $J$ if there is a (not necessarily rooted) homomorphism $h : B \to J$. The effective application of the rule extends $J$ with a new path $P'$ which is a copy of $H$ specialized as follows. The root of $P'$ is $h(\text{root}(B))$. The leaf of $P'$ depends on the type of rule. If the rule is a path inclusion then the leaf of $P'$ is $h(n)$, provided that $\text{leaves}(B) = \{n\}$. If the rule is a mandatory path then the leaf of $P'$ is a fresh node. Every internal node of $P'$ is a fresh node that does not belong to $J$. Figure 1.3 illustrates the application of rules $\sigma_1$ and $\sigma_2$ presented in the previous section, together with the path inclusion $\sigma_3 : \texttt{prof.boss} \longrightarrow \texttt{director}$ stating that the "*the boss of a professor is the director of the department*".

We denote by $Sat(J,\Sigma)$ the saturation of $J$ with $\Sigma$, i.e., the parallel breadth-first application of the rules of $\Sigma$ on $J$ [26]. The notion of query answer then follows, $\mathbf{a} \in \text{ans}(q,J,\Sigma)$ if $\mathbf{a} \in \text{ans}(q, Sat(J,\Sigma))$. As for existential rules, the query answering decision problem in this setting asks to determe whether $\mathbf{a} \in \text{ans}(q,J,\Sigma)$.

The framework we presented naturally translates into existential rules. To a tree $J$ we can assign the existential closure of the formula $\Psi_J = \left(root(x_r) \wedge \Psi_J^{edges} \wedge \bigwedge_i const(l_i)\right)$ where $\Psi_J^{edges}$ is an encoding of the edges of $J$ as binary relations (one predicate is made out of each key of $J$), $x_r$ is the root of $J$ in $\Psi_J^{edges}$ and the $l_i$ are the labelled leaves of $J$ in $\Psi_J^{edges}$. A query $(P,\mathscr{L})$ translates exactly in the same way, except for the fact that when $\mathscr{L} \neq \emptyset$ (that is, we have a get-query) the leaf variable remains free, as it becomes the answer variable of the query. A rule $(B,H)$ is translated as $\forall X\left(\Psi_B \to \exists Z.\Psi_H\right)$ where $X$ is the encoding of the nodes of $B$ as variables according to $\Psi$, and $Z$ is the encoding of the nodes of $H$ that are not in $B$ according to $\Psi$.

This translation is sound and complete with respect to query answering.

**Proposition 34** ([26])**.** *It holds that* $\mathbf{a} \in \text{ans}(q,J,\Sigma)$ *if and only if* $\Psi(J) \cup \Psi(\Sigma) \models \Psi(q(\mathbf{a}))$.

In the definition we noted $q(\mathbf{a})$ for the path query $q$ whose leaf value is labelled according to $\mathbf{a}$. Linking this framework with that of existential rules allows us to import and transfer decidability results between the two settings, as we will do later.

### Decidability of Query Answering

We start by providing an overview of the main results concerning decidability. As already suggested, despite their apparent simplicity path rules embody a great expressivity, allowing one to encode semi-Thue systems [106]. From this we get a first negative result showing that they are a too powerful language for our purposes.

**Proposition 35** ([98])**.** *Query answering is undecidable for path rules.*

This been observed also in other contexts [11, 31, 14], and it can also be shown that undecidability already happens with paths of length two [42]. In light of this, we have to restrict the rule language

expressivity if we want to design a language with decidable query answering. We therefore turn our attention to the special case of two rule languages we call *atomic* constraining the body (body atomic) or the head (head atomic) to be of length one, that is, a path composed of a single key. A rule which is both *body atomic* and *head atomic* is called a *key inclusion*.

**Proposition 36** ([98])**.** *Query answering is decidable for sets of head (respect., body) atomic path rules.*

Query rewriting always terminates with body atomic rules, and a rewriting procedure tailored for this setting can be easily defined [98]. Alternatively, one can also note that body atomic rules translate to linear rules, for which query rewriting is known to terminate. From this, we also conclude that query answering for body atomic rules is in $AC_0$ for data-complexity. For head atomic rules rewriting may not terminate. However, the rewritings produced with these rules will be of non-decreasing size. Hence, because queries are rooted and the depth of the input record fixed by a constant $d$, we can safely neglect all rewritings of size larger than $d$. Beside, the number of steps needed to produce one of such reformulations is polynomially bounded. This allows us to conclude that query answering is in NP for data-complexity. Alternative arguments for decidability can also given by looking at the shape of these rules once translated into existential rules, and exploiting known properties. Concerning head atomic rules, these generate single-head existential rules over binary predicates. These rules are either Datalog rules or rules with a single existential variable which always appear in second position. A specificity of the obtained rules is that they do not allow one to move a fresh null in first position of any predicate. Hence, the translation can be carefully modified (without impacting query answering) and existential variables replaced by constants. Following this scheme we can lower the data-complexity of query answering so as to match that of datalog rules.

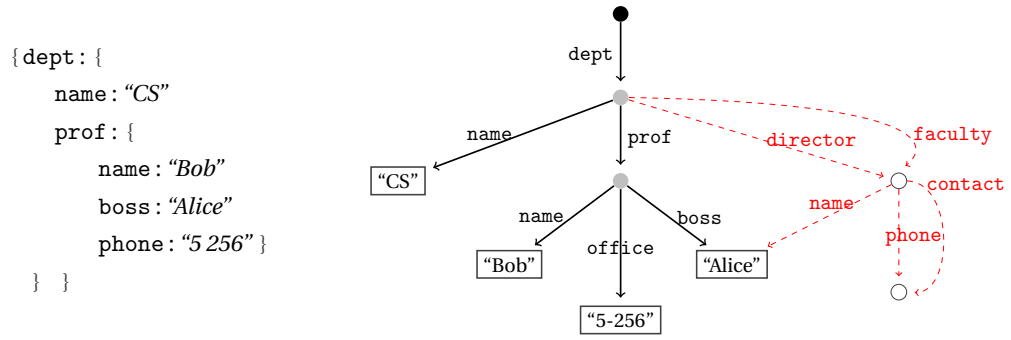
### Contextual and Suffix Path Rules

The next step has been to see how to go beyond atomic rules [26]. Towards this aim, we first equip our rules with contexts. Because JSON records are hierarchical tree structures, a given value can assume a different meaning depending on the context (of the record) where it is employed. From a modeling perspective, this means that it is interesting to dispose of rules with context that apply selectively on the record structure.

**Definition 37.** *A contextual path rule is a pair* $(\sigma, C)$ *where* $\sigma$ *is a path rule and* $C$ *is a (possibly empty) context path, whose leaf node coincides with the root of* $\mathrm{body}(\sigma)$

As pointed out by the definition, we allow the context $C$ to be empty, as for the rules presented before. Moreover, for contextual rules we also allow the body path $B$ of *mandatory paths* to be empty, but at the only condition that the context $C$ is not. This is illustrated by rule $\sigma_3$ of Figure 1.4. This combination can be used to add "default" mandatory paths to *all* nodes within a certain context.

The difference with respect to the applicability of a standard rule is that we require the existence of a homomorphism $h : C \cup \mathrm{body}(\sigma) \to J$ that properly maps the joined context and rule body. Adding




$\sigma_1 : \texttt{dept} : \texttt{prof.boss.\$} \rightarrow \texttt{director.name.\$}$ $\quad$ $\sigma_3 : \texttt{dept.faculty} : \epsilon \rightarrow \exists \texttt{phone}$

$\sigma_2 : \texttt{dept} : \texttt{director} \rightarrow \texttt{faculty}$ $\quad$ $\sigma_4 : \texttt{phone} \rightarrow \texttt{contact}$

Figure 1.4: Suffix rule application

this type of contexts preserves the decidability of query answering for body (resp. head) atomic rule-sets. We however want to move from the atomicity condition, and look for rules that allow for paths of arbitrary length on both sides, but that control recursion differently.

We then introduce *constrained* rules, which feature the capability of constraining the application of a rule to the labelled leaves of the database only.

**Definition 38.** *A* constrained path *rule is a pair* $(\sigma, \mathscr{L})$ *where* $\sigma$ *is a path rule and* $\mathscr{L} \subseteq \text{leaves}(\text{body}(\sigma))$. *We call a path rule such that* $\mathscr{L} = \text{leaves}(\text{body}(\sigma))$ *a* suffix *path rule.*

The difference with respect to the baseline applicability of a rule is that we require the existence of a homomorphism $h : \text{body}(\sigma) \rightarrow J$ that moreover maps (again, the node in) $\mathscr{L}$ to a *labelled* leaf of $J$. Of course, if the rule is contextual, $h$ must also account for the context as defined before. We stress again the fact the leaf must be labelled. It turns out that we can combine the following three types of rules to get a decidable rule language.

**Proposition 39.** *Query answering is decidable for ruleset which are built from any combination of*

*(A) contextual key-inclusions*

*(B) contextual frontier suffix path inclusions*

*(C) contextual mandatory paths*

This language is interesting as it offer several possibilities for writing rules. Type (A) rules define inclusions between keys, hence allowing to organize them hierarchically, and moreover depending on the context as illustrated by rule $\sigma_2$ of Figure 1.4. Such rules can apply anywhere in the tree. Type (B) rules define inclusions between paths ending on a value. They specify multiple ways of accessing a constant value in the record. An example is rule $\sigma_1$ of Figure 1.4. Note that the body of suffix path inclusions use the symbol "$" to denote the application on a constant. Finally, type (C) rules are mandatory path assertions. These allow one to express that some path exists in the structure, even if its extremity is unknown. Example 1.4 illustrates the application of suffix rules. In passing, let us also mention that the first order logic translation naturally extends to the rules we just presented [26].

RELATION TO EXISTING ONTOLOGY LANGUAGES By slightly modifying the FO translation of path inclusion rules (without incidence on query entailment), we see that the rules just presented can be translated into a fragment of existential rules with decidable query answering, namely *frontier-guarded existential rules* [12]. To do so, all suffix rules $\sigma$ are encoded by the rules $\Psi_{\sigma,c_1},\ldots,\Psi_{\sigma,c_n}$ obtained by substituting the variable corresponding to the frontier leaf shared by the body and head of the rule with each constant $c_i$ appearing in the input record $J$. This allows us to recast query answering as atomic query answering over fixed-arity guarded rules, which provides an upper bound for combined complexity, namely EXPTIME [37]. However, since the translation to frontier-guarded rules is not data-independent, this does not yield any useful upper bound for data complexity. Even though the rules we presented are based on paths, their logical translation does not fit into any known DL dialect, due to the presence of contextual key inclusions (rules of type A). Were we to consider only context-free key inclusions, then the preceding translation into frontier-guarded existential rules would be expressible in the DL ELHIO (where role inclusions capture the key inclusions, nominals allow us to speak of constants, and inverse roles are used to capture contexts in rules of type B and C). However, this correspondence does not yield improved complexity results either.

CONNECTION WITH WORD REWRITING SYSTEMS Settled the question of decidability, we are left with the question of the precise complexity of query answering. Our approach to derive tighter bounds is to rewrite an input query into a *word automaton* which represents in a succinct way the (possibly infinite) set of reformulations of a check or get query. More precisely, we show that the problem of computing the rewritings of check and get queries can be reduced to that of computing the (regular) language of ancestors in a an extended word rewriting system [51]. Starting from a set of path rules $\Sigma$, we therefore define a word rewriting system $\Omega(\Sigma)$ which simulates the forward-chaining application of the rules in $\Sigma$.

Before doing that, let us briefly recall word rewriting systems. A word (or string) rewriting system $\Omega$ over a finite alphabet $A$ consists of a finite set of word rewriting rules of the form $(u_1,u_2)\in A^*\times A^*$. We say that a word $\omega$ rewrites to $\omega'$ following the rule $(u_1,u_2)$ if $\omega=\omega_1\cdot u_1\cdot\omega_2$ and $\omega'=\omega_1\cdot u_2\cdot\omega_2$, where "$\cdot$" denotes the word concatenation. We lift this notion to that of extended rewriting systems.

**Definition 40** ([26])**.** *An extended suffix rewriting system $\Omega$ over a finite alphabet consists of a finite set of word rewriting rules of the following form*

- $(C,a,b)$ *where $C$ is a regular language, and $a$ and $b$ two letters.*

  (contextual relabeling rules)

  *Their semantics is a follows: $\omega_1\cdot a\cdot\omega_2$ rewrites to $\omega_1\cdot b\cdot\omega_2$ by such type of rule if $\omega_1$ belongs to $C$.*

- $(C,L,R)$ *with $C,L,R$ regular languages.*

  (contextual extended suffix rules)

  *Their semantics is a follows: $\omega\cdot l$ rewrites to $\omega\cdot r$ by such type of rule if $\omega$ belongs to $C$, $l$ belongs to $L$, and $r$ belongs to $R$.*

[12] these are existential rules in which an atom from the rule body contains all the frontier variables [117]

*We say that $\omega$ rewrites to $\omega'$ if there exists a (possibly empty) sequence of words $(\omega =)\omega_1, \ldots, \omega_n (= \omega')$ such that $\omega_j$ rewrites to $\omega_{j+1}$ by a rule of $\Omega$.*

*The set of* ancestors *of a language $L$ by $\Omega$, denoted by $Anc_\Omega(L)$, is the set of words that rewrite to a word in $L$.*

We need some notation at this point for establishing a connection between query answering and word rewriting. We denote by $\Omega(\Sigma)$ the ERS associated with a ruleset $\Sigma$. We denote by $\hat{q}$ a proper encoding of queries into words for $\Omega(\Sigma)$. Finally, we denote by $q_\omega$ the path query built from a word of the ERS. For more details, we shall refer to Proposition 2 of [28]

**Proposition 41** ([26])**.** $\bigcup_{\omega \in Anc_{\Omega(\Sigma)}(\{\hat{q}\})} \text{ans}(q_\omega, J) = \text{ans}(q, J, \Sigma)$

An important point is that $Anc_{\Omega(\Sigma)}(\hat{q})$ is a regular language. This means that, although the rewriting set of a query may be infinite, we have a finite representation of the set of rewriting of a query by means of an automaton. This regularity result is both key to derive the following complexity results and for the effective implementation of the approach as it opens up for a finite representation of the (possibly infinite) rewriting set of a query.

**Theorem 42** ([26])**.** *QA with key-inclusions, frontier suffix path inclusion, mandatory paths is*

- *in NLSpace for data complexity, with contexts;*
- *PSpace-complete for combined complexity, with contexts ;*
- *NLSpace-complete for combined complexity, without contexts.*

Summing up, query answering can be reduced to (1) computing a regular language corresponding to the ancestors of a word encoding the query in an extended rewriting suffix system associated with the KV-rules (2) evaluating this expression on the KV-store. Note however that the evaluation of regular expressions is not directly supported by KV-stores native languages. Nevertheless, as records have bounded depth $d$, one can still generate all check/get queries whose length is bounded by $d$, thereby making query answering complete. In practice, however, this may be inefficient when the regular expression contains a Kleene-$*$ or large disjunctions, as many queries could be generated. This is a known issue for semi-structured data-management systems, which is tackled by relying on a concise structural summary of the tree, such as data-guides [67]. In our case, by relying on a structure that simply lists all maximal paths in the tree, one can select those belonging to the language of the regular expression, which suffices to ensure complete query answering. The list of maximal paths in a record can be computed off-line in a single traversal of the record.

## TOWARDS TREE QUERIES AND RULES

In the context of the PhD thesis of Olivier Rodriguez, we investigate the extension of the framework towards more expressive tree queries and rules. These results are still unpublished [110], but we present them briefly to outline the trajectory taken by our investigations.

Of course, query answering is undecidable for the case of trees, as it subsumes the case of paths. Nevertheless, the particular question we are interested in is whether an approach based on regular languages can be followed to handle tree shaped queries and rules. Beside the technical contribution, this question makes us also understanding more the extent of the connection with tree rewriting systems, and sharpens the differences which were much less accentuated for the case of paths.

We consider the following notion of query, which generalize that given for paths.

**Definition 43.** *A* tree *query is a triple* $q = (T, \mathscr{L}, \mathbf{x})$ *where:*

- $T$ *is a tree*
- $\mathscr{L} \subseteq \text{leaves}(T)$
- $\mathbf{x}$ *is a sequence of* answer nodes *belonging to* $\mathscr{L}$

*A tuple of* $\mathbf{a}$ *is an answer to a query* $q$ *over* $J$*, if there is a rooted homomorphism* $h : T \to J$ *such that*

- $h(\mathbf{x}[i])$ *is labelled by* $\mathbf{a}[i]$
- $h(n \in \mathscr{L})$ *is labelled by a constant*

Let us now define tree rules. We focus here on the case without contexts.

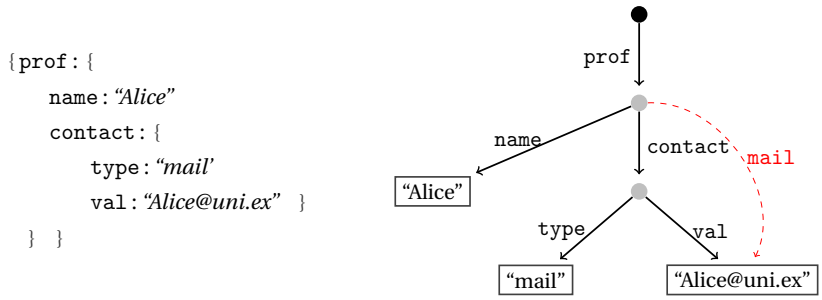
**Definition 44.** *A constrained tree rule is a tuple* $\sigma = (B, H, \mathscr{L})$ *where* $B$ *and* $H$ *are trees sharing the same root and a subset of their leaves, and* $\mathscr{L} \subseteq \text{leaves}(B)$ *is a set of nodes constraining the applicability of the rule to valued leaves of the database.*

*A* frontier-suffix *rule is such that every leaf node shared by* $B$ *and* $H$ *belongs to* $\mathscr{L}$*.*

Extending path rules to tree rules, the notion of rule application naturally follows. This requires the existence of a tree homomorphism $h : B \to J$ that moreover maps every node in $\mathscr{L}$ to a *labelled* leaf of $J$. The application of the rule this time extends $J$ with a tree $T'$ which is a copy of $H$. Again, the root of $T'$ is the image $h(\text{root}(B))$. A leaf $n$ of $H$ becomes $h(n)$ if $n \in \mathscr{L}$ and a fresh node otherwise. As before, all other nodes of $T'$ are fresh.

An example of suffix tree rule application is given in Figure 1.5. The rule $\sigma$ applies on all subtrees representing a mail contact. We denote by $\$_i$ the fact that a (distinct) leaf $i$ of the rule-body must be mapped to a valued leaf in the data ; this value can be copied into a leaf of the tree in rule head.

The first step for accounting for sets of suffix tree rules is to define a sound and complete rewriting operator. Interestingly, there are queries that admit DAG rewritings hence, strictly speaking, that are not tree queries. However, it can be shown that the most general rewritings of a query are all of tree-form and this suffixes to correctly answer queries, as well as for the implementability of the approach.




$$\sigma : \texttt{contact}\big(\texttt{type.``mail''}, \texttt{val.\$}_1\big) \rightarrow \texttt{mail.\$}_1$$

Figure 1.5: Frontier suffix tree-rule application

**Proposition 45** ([110])**.** *For every tree query $q$ and frontier-suffix ruleset $\Sigma$ there is a sound and complete set of tree-rewritings $Q$ such that, for all tree $T$,* $\text{ans}(q, T, \Sigma) = \bigcup_{q' \in Q} \text{ans}(q', T)$.

To compute the set of rewritings of a query, we developed a query rewriting operator based on the notion of *twig-unifiers* [110]. Because of the correspondence with existential rules, twig-unifiers can be seen as a special case of piece unifiers [15]. More precisely, twig-unifiers correspond to the piece-unifiers that give all most general queries, and these are of tree form. As stated below, frontier suffix tree rules can be considered as a staple to define a rule language for reasoning on trees.

**Theorem 46** ([110])**.** *Query answering with tree queries and frontier suffix rules is decidable.*

Naturally this result also follows from the fact that frontier suffix tree rules can be translated into frontier-guarded existential rules. To derive it, we followed however another path. The approach of [26] has been extended so as to define a suitable notion of *tree automaton* capturing the most general set of rewritings of a query [110].

Going through this process the differences between query rewriting and tree rewriting emerged. Query rewriting works on the basis of unification, homomorphisms and logical equivalence. Tree rewriting systems work rather with a notion closer to that of *isomorphism* [51]. Because of this, tree rewriting systems are not adequate to account for redundancies in the case of trees. Note that for the case of paths, the notions of homomorphism and isomorphism coincide, and this is the reason why a clean connection between query rewriting and word rewriting systems can be established [26]. In spite of this, it still holds that the finite state model of computation of tree automata can be adapted to represent possibly infinite sets of query rewritings.

OUTLINE The investigations done so far focussed on building foundations for reasoning on JSON data on top of NoSQL systems. An interesting perspective on this work is that it has been characterized by seeking for connections with other formalisms and transfer results. From one side, we outlined a connection with existential rules, which allow us to see our rules as frontier-guarded ones and directly inherit decidability. These fall in the class of greedy bounded treewidth existential rules,

and widening this work can help at exploring more this class of rules. On the other side, it has been interesting to explore the relationships with word and tree rewriting and better understand their points of contact. As we already said, frontier suffix tree rules can be considered as a very interesting rule language for reasoning on JSON data. There are furthermore several extensions that we can imagine of. First of all, a notion of contexts can be added. Then, a notion of *prefix rules*, applying at the root level instead of the leaves, can be developed. Furthermore, acyclicity conditions can be developed so as to mix frontier suffix (or prefix) rules with general rules while still preserving decidability.

As part of the ongoing work in the thesis of Olivier Rodriguez we are implementing a library for manipulating tree languages, on top of which a query rewriting system for frontier suffix rules can be built [110]. This will allow us to develop and compare the performance of query rewriting using different optimizations. A very preliminary version of the system has been developed for reasoning with key inclusions over trees. In this work a technique for parallelizing the query rewriting process has also been presented [109].

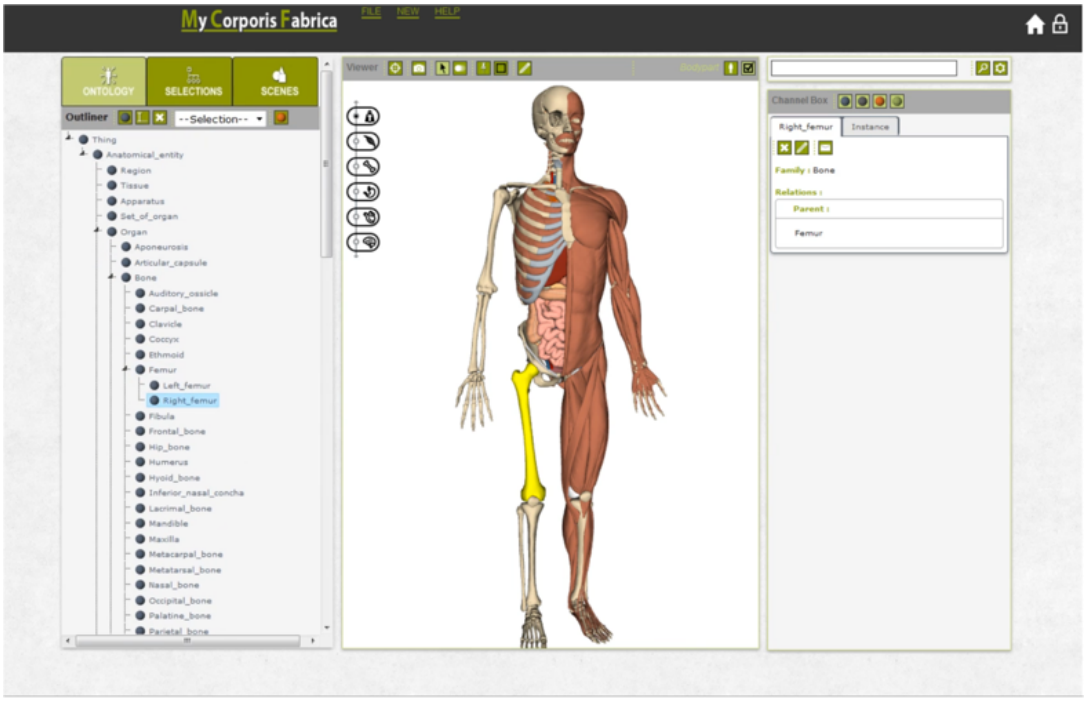



Figure 1.6: The MyCF Browser (video)

## 1.5 Applications of Ontologies and Rules

This last section will presents the main applications of ontologies that we have developed, with a particular emphasis on the lesson learned from these projects. Applications are important as they allow one to both apply existing results and technologies as well as identify new interesting questions that can fuel more fundamental study. We present two multi-disciplinary projects at the frontier of knowledge representation and computer graphics. This work has been done in collaboration with the Inria IMAGINE team in Grenoble, which specializes on image synthesis. The overall goal of these projects has been to use ontologies and rules to enable a better exploitation of 3D models and scenes.

### The My Corporis Fabrica Project

The My Corporis Fabrica project has been led by Olivier Palombi, a neurophysicians affiliated to the Imagine team. The project focussed on using ontologies and rules integrating tools enabling a better visualization and simulation 3D models of the human anatomy [104]. An example illustrated in Figure 1.6 is the MyCF browser [20]. This approach has been extended later to models of the human embryo development [107]. This is a multi-disciplinary project which has seen many participants. From the medical side, Olivier Palombi, Favier Valentin, and Pierre-Yves Rabattu worked on the medical validation of the ontologies. From the computer graphics side, François Faure, Jean-Claude Léon, Damien Rhomer and Benoît Masse worked on the development of 3D models and simulations. Together with Marie-Christine Rousset, we dealt with knowledge engineering and reasoning aspects of the project.

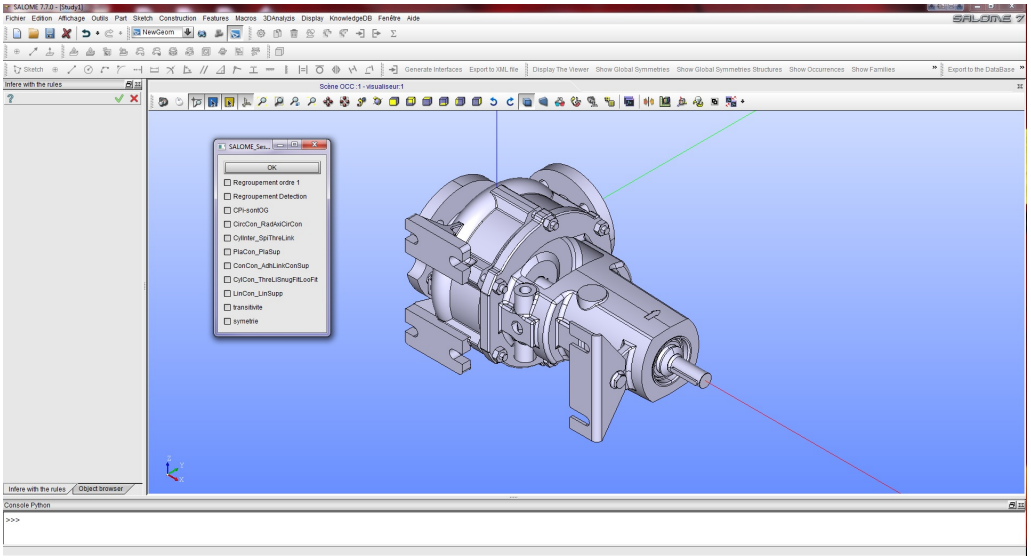


Figure 1.7: 3DAA Module in the SALOME Environment (hydraulic pump model)

### The 3D Assembly Analysis Project

The 3D Assembly Analysis project has been led by Jean-Claude Léon, Professor of mechanical engineering, affiliated to the Imagine team. The project focussed on ontologies for verification and visualization of CAD models 3D of manufactory products [122, 123, 121, 29]. The approach has been prototyped as a plugin within the SALOME software platform developed by OpenCascade, EDF R&D and CEA [7]. This is illustrated in Figure 1.7. This displays a CAD model of an *hydraulic pump* we will use later for examples. In this collaboration, Jean-Claude Léon and Harold Vilmart (funded by EDF) dealt with the 3D and geometry part of the project.

### Lessons learned

The goal of the remainder of the section is to present the main lessons that we have learn from these projects. Our aim is to outline (1) which type of semantic information was needed by the applications, (2) what semantic informations have been useful for and (3) the requirements of rule languages for this applicative use case, and their relationships with the Semantic Web standards.

**Types and Functions** The main concern of both the MyCF and 3DAA projects has been to enable a better exploitation of a collection of 3D models. This goes through the process of labelling the different 3D entities (being solids or meshes) of a scene with semantic information for their *type* and *function*. Here, type and function are intended from the application-domain point of view. Of course the type information allows one to select all objects of a certain *class*. In the anatomical context, this can be the set of bones or tendons within a particular anatomical entity (such as the leg, arm, etc). In the mechanical context, this can be all screws of an airplane wing, or a hydraulic pump.

However, to really make the system interesting from a user perspective, domain experts soon pointed

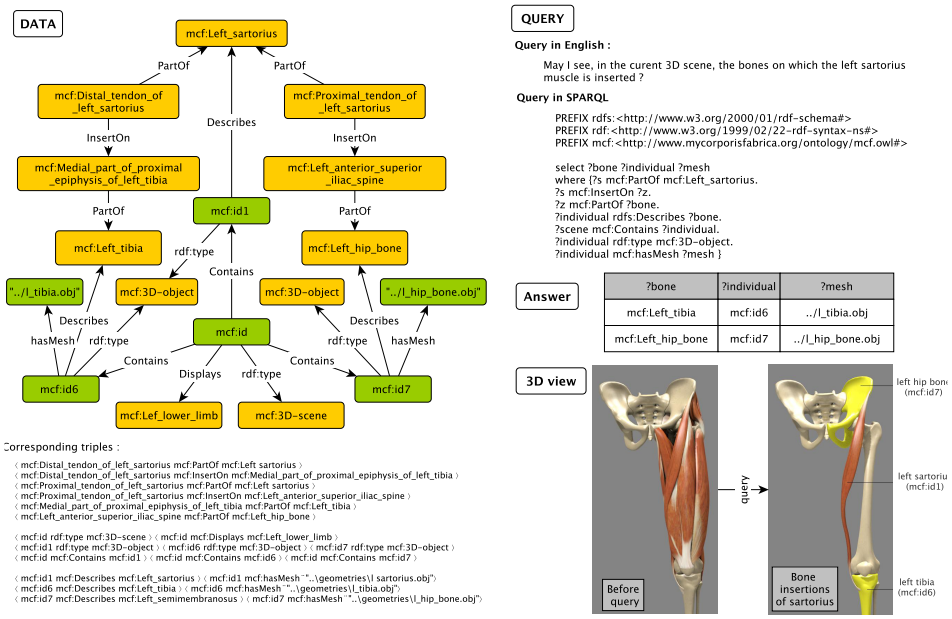



Figure 1.8: Visualization by querying in MyCF

out that taxonomies for classes and properties are not enough. In both the anatomy and CAO context, it is critical to point the *function* of an object, and to be able to query such functions. In the anatomy context, this translate in the possibility of performing a pathological analysis. For instance, being able to select all anatomical entities that participate in a given physiological functions, like the *gait* or *the stability of the knee* makes one able to point anatomical entities that may be responsible for a pathology. In the mechanical context, functions are even richer and range from micro-level functions such as the *planar support* provided by a planar contact between two assembly components to macro-level function such as determining that a set of solids accomplishes the functions of an hydraulic pump (which does liquid flow circulation and pressure generation). This information can be used to assist the CAD editing, as it allows one to retrieve existing objects starting from their functional description to build new assemblies. So, beside type information (classes) functional informations are a really important semantic information that were needed by both applications and knowledge engineering must account for that.

VISUALIZATION BY QUERYING Dealing with 3D models, the best way to report to the user the result of a query is by visualizing it. As 3D scenes can be very complex and articulated, this revealed to be particularly helpful.

Concerning anatomy models in MyCF, one should note that body entities can largely vary in size, and that they can be very close to each other or even hide each other, making their selection difficult.

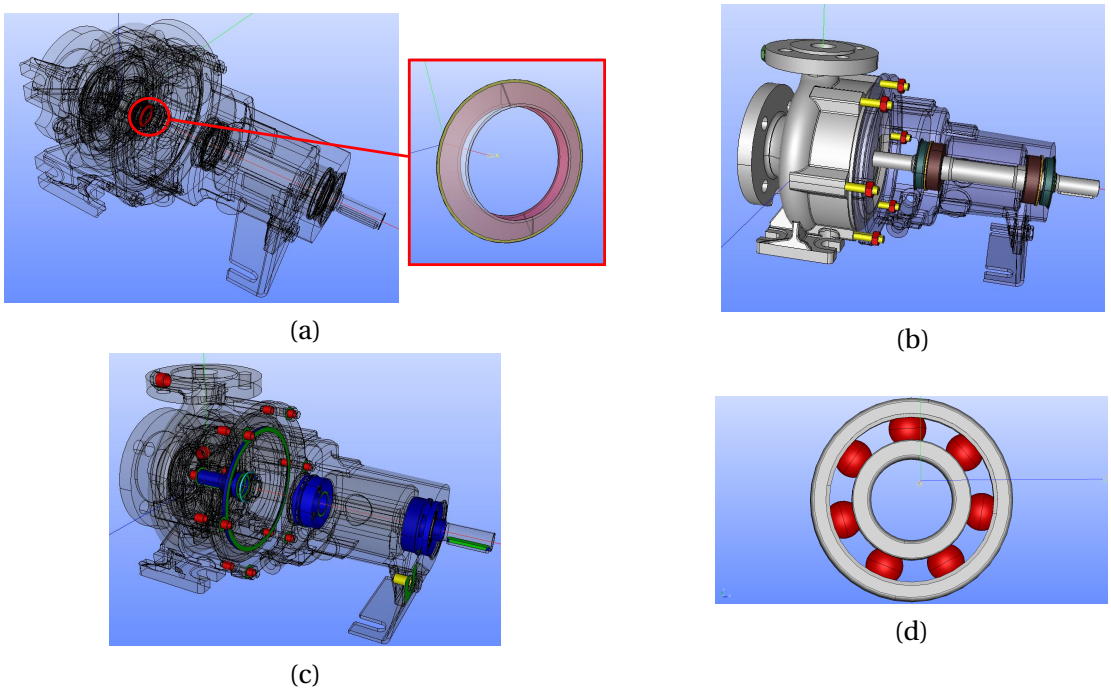



Figure 1.9: (a-c) Visualization by querying in 3DAA: (a) piled-up components (b) groups of solids with same form (c) geometrical contacts and interferences between solids (d) Groupings: example of rolling bearings

Queries leveraging on knowledge and reasoning can provide a much more accurate selection. This is illustrated for instance in Figure 1.8 on a model of the leg. The query result displays the two bones of the leg (in yellow) where the sartorius muscle is attached. Queries can be generated from templates.

It is worth mentioning that the visualization system of MyCF has also been employed to support anatomy teaching at Grenoble university. Beside visualization, queries have been used to select parts of models that can be exported to create simulations [20] of the human body movements using the SOFA tool [61]. The goal was to generate simulations of certain physiological functions such as the gait, the flexion of the knee, etc. Queries made possible the automatic selection (through recursive rules) of the correct set of bones, muscles, ligaments, etc, are required to set up all the 3D objects as well as the mechanical simulation parameters.

For CAD models, the complexity of the scene can even increase when compared with that of the human body (which is a fixed model). In Figure 1.9 several cases of visualization are illustrated. The example (a) shows two stacked Belleville washers, which constitute a so called “piled-up component” which can be difficult to spot in a large model by simply navigating in the CAD editor. Then (b) illustrates groups of solids which have the same topological form (in the picture, the same color is assigned to solids of the same form). Finally (c) illustrates geometrical interferences between solids, as well as planar and cylindrical contacts between solids. Note that in all of the cases one needs a module which processes the results of the queries and make them more readable at the interface level. This is standard for computer graphic applications.

Finally, queries can be used to verify the coherence of models, notably verifying that certain configurations between solids that are implausible either from the anatomical or mechanical perspective do not arise. This feature has been leveraged in both the MyCF and the 3DAA project.

Summing up, a rich semantic annotation of 3D models, and in particular concerning the function of objects, opens up for the selection of components based on complex queries. This can serve as the basis of visualization, preparation of simulations, or verification of the model coherence with respect to the domain knowledge.

KNOWLEDGE ENGINEERING The MyCF and 3DAA projects started with the will of being compliant with Semantic Web technologies and leverage Triplestores such as Jena [46] and Virtuso [8] for storing and querying data. And indeed, both the MyCF and 3DAA *factbases* have a proper RDF representation. Nevertheless, to meet the application requirements, the *rulebases* developed for both projects did not fit the typical OWL 2 fragments.

*The case of MyCF.* The Semantic Web promotes the *reuse* of publicly available ontologies thereby capitalizing on the efforts made to build knowledge bases. Accordingly, the MyCF ontology reuses the FMA ontology, which is considered a reference model for anatomy [111]. MyCF inherits from FMA a taxonomy of 70.000 anatomical entities. This is however far from being sufficient for exploiting 3D models of the human anatomy. MyCF needs to add *part-of* relations between anatomical entities (saying for example that the sartorius muscle is part of the leg) complementing *subclass-of* relations (indeed, the sartorius muscle is *not* a subclass of the leg). Further, MyCF needs functional entities, such as gait, breath, and stability, that denote the functions of the human body, and are the fundamental knowledge to explain the role of each anatomical entity. Hence, 4000 physiological functions had to be added, and anatomical entities had to be linked with them (this has been done manually by physicians involved in this project). Finally, on top of this, 3D scenes modelling patient-specific anatomical entities had to be linked with the proper anatomical entities. So, important additions had to be done to the ontology fragment imported from FMA. However, importing an existing ontology obliges one to embrace the modelling choices that have been made with it. In this case, the crux is that every anatomical entity in FMA is represented by a class.[13] This made the extensions required for MyCF possible only using meta-modeling (also called punning) where classes can also be seen as terms (individuals of meta-classes). This is illustrated for instance in Figure 1.8 where all yellow boxes represent classes which are linked by properties such as *part-of* and *insert-on*. This modeling evades from first-order logic interpretations and hinders the use of reasoners with a first-order semantic (those for OWL in particular). The language OWL 2 full (which is undecidable) allows for domain-metamodeling but there is no reasoner for it. The approach here has been to keep the dataset in RDF and add datalog rules over RDF triples, akin to RDF deductive triplestores [98]. RDF being a simply exchange format for triples make that Triplestores do not reject this encoding of the factbase (while an OWL reasoner would). The resulting knowledge base contains a set of datalog

[13] In other ontologies, like Gene Ontology [13] terms are rather used to represent genetic entities, which makes extension much simpler.

rules over RDF triples, mostly akin to transitivity and property chains. The Jena RDF engine has been used for doing reasoning [46].

A similar situation arose in the follow up project concerning the development of the MyCF Embryo ontology [107], which inherited from the human developmental anatomy ontology EHDAA2 [19]. Overall, in spite of this incongruence with the most widespread Semantic Web ontology languages like OWL, query answering is evidently decidable here as reducible to datalog reasoning. We believe that domain-metamodeling occurs in many applicative cases similar to the one just described and we believe that extending OWL reasoners to support these cases would be helpful for seamlessly reusing ontologies [92].

*The case of 3DAA.* In this project the aforementioned problems have not been encountered, because a novel ontology model for assemblies and manufactory products has been created from scratch. To define a bit better the goal of this project we precise that, in contrast with the MyCF project, where all 3D models have been manually linked to anatomy concepts by medical experts, the goal of the 3DAA projects was to propose an *automatic* method for inferring semantic annotations for solids in a 3D scene, using a reference ontological model. This is done by a two phase approach. In the first phase, a CAD model is analyzed from the geometric point of view with a geometric modeler such as SALOME [7] so as to extract an RDF factbase describing the solids it contains, their interfaces, and their properties. In the second phase, this factbase is enriched by reasoning. The originality here is that rules are used to compute inferences that would be cumbersome to implement in the geometric modeler. Thus, the computation is balanced between two complementary systems each focussing on the tasks that they can handle at best (being it geometry or reasoning). The specificity of the ontology vocabulary here is that it uses only binary relations. The knowledge base contains a taxonomy of 300 classes to which 100 inference rules are added. The rules contain existentially quantified variables and multiple frontier variables. Because of this, they are best expressed as existential rules, and do not fit any OWL fragment. Moreover, rules can feature from 10 to 20 atoms in their body and head, and this fragmentation is due to the choice of working with binary properties and from the articulation of the ontology model. We believe that this fragmentation is a common problems when expressing a complex business logic over binary predicates. This shows the interest of having higher arity predicates, and raises the interesting question of optimizing reasoning in this kind of situations.

Finally, this project also shows that existential rules need extensions to be practically useful. The application called for the use of negation and inequalities in the rule body. Moreover, the application showed that it would be interesting to extend rules beyond a first order logic setting. An interesting example is that of having *set-variables* [86, 45] to capture slightly more sophisticated mechanical objects. Figure 1.9.(d) shows the example of the rolling bearings. These are defined as “maximal sets of spheres, all being in contact with the same internal and external rings”.

To conclude, knowledge engineering is an important part of the process of deploying applications based on ontologies. Standards constitute a valid starting point for building ontologies, but can also bring constraints that the deployment of real applications needs to overcome. We advocate that existential rules and their extensions are a valid framework for building applications.

# 2 Research project

We conclude this dissertation with a presentation of our vision of the forthcoming research activities.

Project Implementation The presented research program will be carried on in the context of several actions in which I am involved with a principal role. I lead the ANR CQFD project (2019-24) and I will lead the Inria team BOREAL (creation planned for January 2022), which will be the follow up of the Inria GraphIK team, of which I am currently a member. Both projects strongly commit to the investigation of reasoning on *heterogeneous and federated data* - the topic I will be targeting. Another important initiative is the participation in the Inria-DFKI Project R4Agri (2021-24) focussing on reasoning on heterogenous data for Agriculture. Finally, I lead the Inria ADT project proposal (Aide pour Développement Technologique, Inria engineer funding) and supervise the development activity for a major version of *Graal*, a library for reasoning on heterogeneous and federated data.[14] In parallel with these activities, I have discussions and exchanges with companies interested in rule-based data integration to seek for synergies between industry and research on the topic of reasoning. The project that will be presented will therefore draw from a long term vision of reasoning on data englobing all of these initiatives.

## Knowledge and Rule-based Languages for Exploiting Heterogeneous and Federated Data

Facing The Data Variety Issue Current information systems are grounded on the exploitation of data coming from an increasing number of sources. Handling the variety of data becomes a more and more challenging issue for enterprises and institutions willing to get new insights from their data.

To illustrate, it is well known that the most challenging issue for data-scientists is that of accumulating high quantities of high-quality data to run their mining and analytic tasks. In reason of this, it has been regularly reported that they spend most of their time gathering, cleaning, preparing and organizing data from numerous sources.[15] In many contexts, it has also been recognized that the bottleneck for exploiting a data source is the translation of the (unstructured) domain-expert questions into sets of (structured) database queries or programs [91]. This process can take up to days depending on the organisation, largely encompassing the time of running queries or data analysis task themselves. Finally, errors in data or data processing over multiple databases can lead to bad analysis supporting wrong decisions thereby raising the urgence for tracing data, and justify and explain answers to queries.

As an illustrative case of data-variety, let us consider the MIMIC project for healthcare [79]. MIMIC is one of the largest electronic patient record datasets. It has been released to the scientific community by the Beth Israel Deaconess Hospital in Boston with the aim of promoting the creation of tools

[14] The proposal has received a very positive evaluation. For administrative reasons, the funding has been postponed after the creation of the BOREAL Inria team. In spite of these delays, the development started in fall 2020 with other credits.
[15] The 2017 and 2018 Data Scientist Report, CrowdFlower.

for preventing mortality in hospital intensive care units. The case of data variety here is well illustrated by Stonebraker in a seminal article on the topic of heterogeneous and federated data [116]. MIMIC contains data acquired from autonomous databases, which have been deployed by independent units of the hospital. For example, it contains *i*) patient meta-data (such as date of birth, ethnicity, admission date) retrieved from structured data sources, *ii*) time-stamped data sourced from bedside activities like diagnosis, medications, and other clinical procedures obtained by monitoring the patient's activity and *iii*) progress notes written by doctors and nurses combined with prescription information drawn from semi-structured and unstructured data. Not only MIMIC data is diverse, but at the moment of exploiting it, users want queries which may (1) draw each time from a different source or (2) cross query multiple sources at once. For example, before analyzing clinical timelines issued from a given source, target patients can be selected by considering their admission records or clinical profiles from another database [50, 62]. There is therefore the need to integrate all of this data in a unified view to allow for its exploitation, and this comes at a cost.

Following Lenzerini [91], let us outline some of the main issues when facing the data variety issue. To to implement a *data-service Q* (here intended as a query or a data-centric program) one should cope with several challenges.

- Data sources relevant for *Q* may be *heterogeneous*. That is, they may exhibit high differences with respect to data models, languages, and the systems that are actually in use. Not to mention the fact that data can differ also with respect to the level of refinement, certainty, and dynamicity. The heterogeneity issue is not new per se [126, 125], but it is magnifying of importance in our times, as informations is produced everywhere, and there is an increasing interest in integrating disparate data. With the diversification of the nature of data and the possible type of target applications the available data management solutions are constantly growing and evolving. However, in spite of heterogeneity, *one must be able to implement a data-service Q on each type of data source.*

- Data sources relevant for *Q* may periodically undergo a corrective maintenance, and more generally through evolutions, thereby departing from their initial design [91]. Evolutions are triggered by new applicative requirements. Indeed, databases are always coupled with applications, and when applications evolve, databases must follow. Concretely, typical operations consist at altering the data structures (being them relations, records, graphs) their fields, their data types, as well as integrity and dependency constraints. Operational databases may be huge and have for example from hundred to thousands tables [80]. The result is that *"the data stored in different sources and the processes operating over them tend to be redundant, mutually inconsistent, and obscure for many users"* [91]. What should be observed here is that *maintaining a data-service Q over a data-source is an unavoidable, time-consuming, and code-degrading activity.*

- Data sources relevant for *Q* may need to be cross-queried. As organizations grow and ramify in different departments and units, it is inevitable that their information systems become more and more scattered into autonomous databases, each designed, deployed, and tuned, for a specific purpose. This is well illustrated by the example of MIMIC. The same holds also for scientific datasets, when the problems of interest scales, the datasets begin to multiply. This however creates a per-

petual tension with decision making process which requires information to be reunited. Moreover, when multiple autonomous databases are cross-queried or interoperating, *explanations* for query results become more and more important to point out the reason for errors.

For all these reasons, providing a data-service in a data-variety setting is a significant challenge.

*On the user side*: it can be difficult for domain-experts and data-analysts to retrieve relevant information without knowing data storage details, and more specifically translate their analysis questions into structured database queries because of the complexity of independent frameworks and languages. This makes domain-expert users dependent on IT-experts for accessing data [91].

*On the administrator and data curator side*: it can be time-consuming to implement data-services on top of independet heterogeneous sources, implement data-services that include multiple sources, maintaining data-services after source corrective maintenance, export and reuse data consistently, provide justification for results of queries and point out errors. This results in data-services that are coded and maintained across sources, which is costly to deploy, maintain, and debug.

These are among the main barriers for the exploitation of heterogeneous and federated data. Hence, we believe that these issues calls for integration paradigms allowing one to better understand data, mechanisms for automatically accessing and querying that adapt to the different types of sources, as well as declarative languages to drive the whole data processing.

### KNOWLEDGE-BASED DATA MANAGEMENT

In this research project, we pursue the Knowledge-Based Data Management (KBDM) approach to attack the grand challenges posed by the exploitation of heterogenous and federated data. The idea of KBDM is to orchestrate access to information through a three-layer architecture, which is common to data-integration [90] and Ontology-Based Data Access (OBDA) [105].

This is constituted by a data-source level, a mapping level, and a knowledge-base level, as illustrated in Figure 2.1. The data level contains the databases to exploit.[16] At the opposite, the knowledge-base (KB) level is the one which is *i)* closer to the users and *ii)* where the business logic is performed through reasoning with expressive rule languages. The mapping layer lies in between: its purpose is to select and transform the data populating the KB. The KB is built around a conceptual model (or ontological model) of the application domain to which conform both the factbase and the rulebase. The factbase is instanciated by the mappings. Importantly, the rulebase considered here is a set of *existential rules* (and extensions thereof) which can model not only ontological knowledge and database constraints, but more generally declarative data processing needed by the applications.

What really characterizes KBDM is the idea of leveraging on (1) *formalized domain-knowledge* and (2) *rule-based languages* to effectively exploit data. Let us explain the importance of both.

[16] At the source level, the heterogeneity of data can be qualified in many aspects. It is important to mention that the main focus here is on the integration of structured and semi-structured data. Unstructured data like text is out of scope.

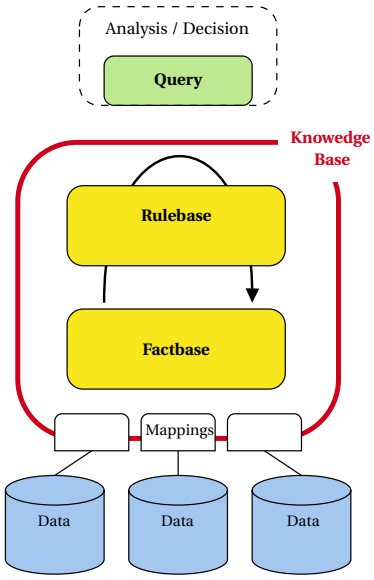



Figure 2.1: Architecture of a KBDM System

KNOWLEDGE. Leveraging domain-knowledge allows curators to bind source data with concepts and relations of the ontological vocabulary of the KB. This brings a very strong semantics for the raw data. By "strong semantics" here we mean that the semantics is both related to the application domain (hence meaningful for domain-experts) and logical (hence it enables reasoning). This paves the way to a more principled reuse of data, with benefits for both users and data curators (administrators). First of all, this allows one to palliate to the aforementioned difficulties in accessing data. Concretely, domain-experts and data-analysts can formulate a high level query at the conceptual level using the ontological vocabulary, which frees them to know data storage details. This makes them less dependent from IT-experts in translating their questions into queries. Evidently, making users independent from IT-experts in exploring data shortens the analysis time. But, more generally, allows for a wider reuse of information by opening up the information system to more users [91, 105].

Another net advantage of using mappings is that queries and data-services expressed at the KB level are *insensitive* to changes in the design or the number of the underlying sources. This is very important for mitigating the costs of maintaining data services. Typically, when one source evolves all data-services using that source must be maintained. In KBDM, a data-service is rather expressed at the conceptual level, and then *automatically translated* to every source of interest by using mappings. Therefore, the maintenance activity is circumscribed to the mapping level - while the data-services are left untouched. It is also important to notice that building on a domain-ontology is key to guide the specification of data services. Indeed, replacing domain knowledge with a model too closed to the raw data could degenerate again to multiple implementations of the same data service (although this time at the conceptual level) taking with it all the issues we mentioned before.

Summing up, leveraging knowledge provides a strong semantics which leads to a better access and reuse of source data and eases the maintenance of data services - and even more if facing data variety.

RULE-BASED LANGUAGES. Existential rules (and extensions thereof) can be seen as a uniform high-level declarative language to perform data processing, which moreover opens up for explainability of queries and traceability of data. It is well acknowledged that declarativity allows for greater modularity and reusability. For instance, using rules mitigates one of the problems of deploying data-services on domain-specific applications, which is that of encoding domain-knowledge inside each data-service implementation. Existential rules settle the basis for very expressive data-processing which, despite sharing the same logical foundations, can go beyond queries mediated by ontologies or database constraints. A very recent example illustrating recursive queries for analytics is [72]. Finally, rule based reasoning supports explainability by design. This is particularly helpful when needed to understand wether a wrong query result is caused by the data or by the processing.

Summing up, the result of adopting the KBDM approach is that of having a unified view of a collection of sources, in the spirit of data-integration, where domain-specific knowledge and mappings set the basis for data access, and rules are the explainable declarative language driving data processing.

KBDM AS FEDERATED SYSTEMS The classic approach for handling heterogeneous data consists at centralizing all information in a datawarehouse with ETL-style (Extract, Load, Transform) procedures. However, the focus of this projects leans more towards *federated systems*, that are middleware abstracting over a set of autonomous databases. One of the main interests of federated databases is the possibility of integrating diverse sources without necessarily moving the data or changing the platforms (although, as we shall discuss next, this approaches can be mixed with materialization in KBDM). By leaving data at the level of sources, it is possible to quickly integrate autonomous system without embarking on the expensive creation and deployment of a datawarehouse. Delegating data processing to the underlying autonomous systems also allows each database to perform at best on the tasks it has been originally deployed and tuned for (for example a NoSQL system may perform better at querying JSON data than a datawarehouse integrating the same type of data). Federated systems also allows one to better deal with frequently updated sources as these hold the latest fresh copy of data.

CONVERGING EXISTING PARADIGMS KBDM is the convergence of well known paradigms for exploiting data such as data-integration, ontology-based data access, and ontology-mediated query answering. With respect to data-integration (and data exchange) OBDA considers that the notion of *target schema* (typically featuring functional and inclusion dependencies) is replaced by that of an ontology [91]. With respect to OBDA, the KBDM approach wants to stress the fact that the rule base allows one to define possibly very complex data processing, on top of an ontological vocabulary. As a matter of fact, OBDA primarly focussed on rulebases made by Description Logic ontologies supporting virtualization, notably DL-Lite [41]. In KBDM, the aim is to study existential rules, as well as the extensions of this formalisms called by data processing.

ON TOP OF DATABASE SYSTEMS An important factor of success for OBDA systems has the possibility of deploying them on top of existing database systems developed and optimized over many decades. The same holds for KBDM. There is a very large number of efficient SQL, NoSQL, graph-based, solutions that are available and provide extremely performant access to data. As such, the focus of this project will be on reasoning on top of existing databases and on best reusing them. It is also worth noting that a new breed of multi-store systems (also called polystores) has emerged during the last years [103]. These systems tackle the problem of efficiently accessing data specifically in federating settings by optimizing query and data access. Polystores add to the critical mass of database technologies available to build efficient federated KBDM systems that allow for intelligent access to data.

DIALOGUING WITH ML SYSTEMS It is not rare today to see a tendency in opposing reasoning and machine learning approaches. Yet, many advocate that the future will be in the combination of the two [68]. Although this is not the main focus of the research project presented here, let us mention that KBDM systems and machine learning systems can dialog and support each other. We foresee at least three types of interactions. The first, is to use KBDM systems to integrate and prepare the data needed by machine learning tasks. The second, is to use machine learning algorithms to generate data which feeds a KBDM system. The third is to inductively learn rules for reasoning, or learn decision trees that can be converted as rules, which again can feed a KBDM system. Making calls to machine learning algorithms (as if they where external services) during forward chaining reasoning is also among the possibilities, but it requires to extend the expressivity of the typical rule languages.

## RESEARCH PROGRAM

The study of KBDM systems with existential rules and their extensions opens to really many issues. In this section, we shall present the questions that constitute the principal focus of our attention for the next years.

- KBDM needs a study of architectures exploring the configurations of the three layers of the systems namely sources, mappings and knowledge base, best suiting applicative use-cases.
- KBDM needs foundational work for better understanding the decidability and complexity tradeoffs of reasoning with queries, rules, constraints and mappings.
- KBDM needs efficient algorithms for query answering based on approaches mixing materialization and virtualization, and leveraging more on the data-source capabilities.
- KBDM needs means to introspect the system in place, in particular concerning the traceability and explanation of answers to queries.
- KDBM needs implementations and testing assessing the efficiency of the proposed techniques, and more generally a platform for realizing software developments and experimental analysis.

### Architectures and Mapping Languages

The realm of possibilities in heterogeneous and federated data management calls for a study of KBDM architectures ranging over several configurations. This includes the case where the system accesses a single source (through multiple mapping assertions) to the case where a federation of independent sources is targeted, to the more complex case of multi-level architectures, where KBDM systems progressively refining information are stacked and connected through expressive mapping layers (i.e., one KBDM system is seen as a source for another KBDM system) in order to achieve complex data manipulations.

When designing a KBDM architecture, an important question is to define the mappings required for integrating heterogeneous data. A mapping is a data-access directive expressed according to the API of the source, then used to instantiate some relations of the vocabulary of the knowledge base. From the data-model point on view, we are interested in considering mappings allowing one to handle data differing in content, format, and platforms, in particularly for federations of NoSQL systems. However, beside structural heterogeneity, also the refinement of data pose interesting challenges.

Motivated by ongoing collaborations with INRAE (French National Research Institute for Agronomy) and the R4Agri project (2021-2024) with DFKI (German Institute for Artificial Intelligence), we plan to study the mappings needed to integrate data which differ in terms of refinement. Taking agronomy as the application domain, we realize that the spectrum of data refinement can vary from raw sensor data, to operational data, to "smart-data" produced with added value provided by experts. Integrating smart-data, can imply the manipulation and transformation of complex objects. A concrete example we are concerned with is that of production and transformations itineraries, found in many INRAE applications [30, 77]. Integrating such types of information may call for more expressive mapping languages, in particular enriched with recursion. At the opposite side, in a project with DFKI we find that handling time-stamped data produced by sensors on a field may require aggregations over time windows in mappings as well as adding so called interpretation capabilities [73]. Interpretation capabilities make that the mapping result is possibility specified as a function of the knowledge base (in addition of course of the source data), that is, data injected in the knowledge base depends on previous content of the knowledge base. This lifts the expressivity of mappings, and raises interesting questions in terms of reasoning and implementations.

### Foundations of Rule Languages

The main source of expressivity of a KBDM system comes from its rulebase. In line with the research we presented at the beginning of this dissertation, the analysis and design of rule languages for reasoning on data is an issue to which we shall pay particularly attention. First of all, we will continue investigating the fundamental decision problem for existential rules, notably query answering, chase termination, boundedness and First-Order rewritability. Several open questions have already been presented in Section 1.3. We will also continue seeking for decidable extension of rule languages for reasoning with tree-like data. This project has been motivated by the work on JSON databases

presented in Section 1.4, but it actually can also be seen as a way of studying fragments of the gbts (greedy bounded treewidth) class of existential rules [119]. Then, going towards more advanced data processing needed by KBDM, an important research direction will be the extension of existential rules with capabilities that are really needed by applications manipulating data such as functions, builtin-predicates, stratified-negation, and aggregates. Among these extensions our interest will be in particular for aggregates and set variables, whose practical utility has been illustrated by the 3DAA project in Section 1.5 for instance. These features are known for sensibly increasing the complexity of the reasoning tasks, and the goal here is to import existing results and study practically useful extensions of existential rules that still retain decidable query answering.

### Algorithms and Optimizations for Query Answering

Reasoning opens up for a better exploitation of data. However, the expressivity of rules makes also computation more involved. Reasoning therefore requires optimizations. We present the approach that to us should be followed to make KDBM reasoning efficient from an algorithmic standpoint.

**Data and Mappings in-the-loop** Moving from the KB to the KBDM setting we add the dimension of data, and with it that of mappings. In KBDM optimizing reasoning calls for studying the interplay between queries, rules, mappings, and data. Indeed, it has been shown that, even in the presence of a single source, reasoning should not be agnostic to the underlying databases [32], and rather take advantage of their capabilities to maximize the performance gainings. This generally helps at further pruning the rewriting set of a query or to limit the materialization phase by avoiding multiple integrations of the same information. This issue has been investigate in the OBDA setting for DL ontologies, and our goal is to investigate it for the case of existential rules which has been little considered so far.

**Physical-Level Reasoning** Rule language express data processing at the conceptual (or ontological) level. In analogy with the evaluation of queries in database systems, conceptual-level reasoning should however be decoupled from physical-level reasoning. As for databases, it should be up to the reasoner to determine its own physical data-access strategy. Although this seems evident, it is exactly the opposite of what is done for instance by common implementations of the chase, where rules expressed at the conceptual level are applied *as they are*. Hence physical-level reasoning is mirroring conceptual-level reasoning. From a performance perspective, this could lead to suboptimality by (1) applying useless rules, (2) computing too many joins, (3) slowing down the update of the knowledge base. The issue of (1) is long known and led to strategy like semi-naive evaluation [10], GRD-driven evaluation [15], datalog-first [87] to mention few. The issue of (3) has been studied recently in [120]. We plan to focus on (2) and in particular on the reshaping and composition of rules for generating data layouts and reasoning-plans able to speed up the task.

**Self-tuning Reasoners** Although deciding non-trivial properties of existential rules is generally undecidable, the amount of sufficient conditions developed ensuring termination of forward or back-

ward chaining made us progress in the understanding of logical reasoning within this language, and we argue that this constitutes a solid ground to handle a large portion of rulesets found in real applications. It is time to go further in the refinement of *self-tuning reasoners* that are able to chose a complex strategy depending on the input data and rules. Surprisingly, there are very few proposals that go in this sense. Toward this goal, the first is to design and choose efficient combined approaches for query answering [93]. Such approaches rely on judiciously mixing materialization, virtualization, saturation and reformulation. In passing, let us mention that, beside the methodological interest, this is also a need encountered by one of the companies using rule-based engines we are discussing with.

FACTORIZED REPRESENTATIONS Finally, we plan to consider the use of factorized representations of materialized facts or rewritten queries, introducing knowledge compilation techniques for rules [83] as well as features of regular languages [26]. This can be helpful to approach the gbts class of existential rules [117], whose models feature repeating patterns which we already show are found when reasoning on NoSQL databases. This can allow one also to go beyond conjunctive queries as the main language for query rewriting, but also fact saturation, thereby storing compact representations of the chase result.

### EXPLANATIONS

Justifying the results of queries serves to achieve a better understanding of data and processing. The need for justifications even increases when federating multiple systems. We choose to illustrate this towards the biais of an industrial application of rules (edited by a company we are discussing with). This consists of a data-service classifying accountancy expenses (belonging to one among several classes). Data comes from different operational databases and classification is done by several hundreds of rules. Importantly, roughly half of the rules are written by hand by experts, while the remaining half have been inductively learned. Despite the efficiency of the automatic classification system in place, in this context it is paramount to be able at any time to justify the classification outcomes. First, it is important to outline the provenance of the data, notably the origin (data-source) and the time of import. Indeed, the data sources of the company may be more or less complete and trustworthy in their data export at a given time. Then, it is important to explain the sequence of rules that supports the classification, and show the contradictions (especially when these concern conflicts between manually edited and learned rules). Indeed, when finding an unexpected answer to a query the critical problem is to know whether this is due to a problem in the *data* (and hence under the responsibility of the entity produced the data) or in the *system* processing the data (and hence under the responsibility of the entity who edited the software). Hence, having mechanisms for explaining answers to queries, tracing the original data needed for answering the query is key to attack this kind of problems. In all of applicative contexts like this one, correctness of information largely comes before fast processing. We plan to import and investigate existing provenance-based frameworks [115] into existential rules. The first step is to use metadata (origin, subjects, confidence, quality, time va-

lidity, etc) to ensure the traceability of facts in the knowledge base. Second, the system should be able to handle explanation questions, asking why an answer is a result to a query or why a result expected by the user is not an answer to a query. In a KBDM system, the additional difficulty is that relevant data is not at the user level, hence mappings have to be incorporated in the explanation process. We believe that these features are key for making existential rules a valid declarative data-processing language from the practical perspective.

### SOFTWARE DEVELOPMENT

The foundational and algorithmic study of KDBM systems we target will be complemented by a software development activity pursuing the ambitious goal of developing and maintaining an open-source Java library for KBDM with existential rules (and extensions thereof). This project is related with the ADT project we mentioned at the beginning of the section, and builds on the Graal library [2] which is the main software of the GraphIK team. The current version of Graal focuses on reasoning over existential rules knowledge bases, and the goal is to prepare and maintain a major version of this tool tackling the issues raised by heterogeneous and federated data. The new version of Graal, will be designed to be a general platform for implementing and testing algorithms and optimizations for KBDM. Beside the fact that the tool will target *efficiency* by including state of the art techniques for reasoning on data within KDBM systems, there are several points that make this project distinctive.

(*Genericity*) the tool will be designed to allow one to implement and compare different reasoning techniques - in the same platform. This is important for accelerating prototyping and doing meaningful experimental comparisons, by isolating specific factors in order to explain performance shifts.

(*Reusabilty*) by capitalizing on the experience built in the development of the first version of Graal, an important goal is to make the new version a glassbox by working on the readability of the code, its extensibility and reusability, which is a necessary feature for attracting users willing to develop novel extensions, but also to ease the long term maintainability of the tool.

(*Expressivity*) None of the current available tools is targeting efficient algorithms for existential rules with heterogeneous and federated data, although proposals for existential rules (e.g., VLog [43], RDFOx [100]) or heterogeneity (e.g., Ontop [39]) or federation (e.g., Corese [52]) already exist.

It is worth mentioning that the project started in fall 2020 with the hiring of Florent Tornil as an engineer working on the next version of the Graal tool. Not only having such a tool will constitute a scientific contribution, but it will also be instrumental in building novel applications and establishing new collaborations both at the industrial (with the companies we are discussing with) and research level (within projects like CQFD and R4Agri).

# REFERENCES


[1] (Software) CouchDB. couchdb.apache.org.

[2] (Software) Graal. graphik-team.github.io/graal.

[3] (Software) IBM Informix. www.ibm.com/products.

[4] (Software) MongoDB. www.mongodb.com.

[5] (Software) Oracle NoSQL Database. nosql.oracle.com.

[6] (Software) PostgreSQL. www.postgresql.org.

[7] (Software) SALOME. www.salome-platform.org .

[8] (Software) Virtuoso. virtuoso.openlinksw.com.

[9] Serge Abiteboul, Omar Benjelloun, and Tova Milo. Positive Active XML. In *PODS*, 2004.

[10] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

[11] Serge Abiteboul and Victor Vianu. Regular path queries with constraints. *J. Comput. Syst. Sci.*, 58(3):428–452, June 1999.

[12] Miklos Ajtai and Yuri Gurevich. Datalog vs first-order logic. 1994.

[13] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[14] Jean-françois Baget. Représenter des connaissances et raisonner avec des hypergraphes: de la projectiona la dérivation sous contraintes. *Université de Montpellier II. Phd Thesis*, 2001.

[15] 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.

[16] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI 2011*, pages 712–717, 2011.

[17] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 712–717. IJCAI/AAAI, 2011.

[18] Pablo Barceló, Gerald Berger, Carsten Lutz, and Andreas Pieris. First-order rewritability of frontier-guarded ontology-mediated queries. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1707–1713, 2018.

[19] Jonathan Bard. A new ontology (structured hierarchy) of human developmental anatomy for the first 7 weeks (c arnegie s tages 1–20). *Journal of anatomy*, 221(5):406–416, 2012.

[20] Armelle Bauer, Federico Ulliana, Ali-Hamadi Dicko, Benjamin Gilles, Olivier Palombi, and

François Faure. My Corporis Fabrica: making anatomy easy. In *SIGGRAPH Studio*, pages 16–1, 2014.

[21] Robert Baumgartner, M. Ceresna, Georg Gottlob, M Herzog, and V. Zigo. Web information acquisition with lixto suite: a demonstration. In *ICDE*, 2003.

[22] Robert Baumgartner, Sergio Flesca, Georg Gottlob, and Christoph Koch. Visual web information extraction with lixto. In *VLDB*, 2001.

[23] Bartosz Bednarczyk, Robert Ferens, and Piotr Ostropolski-Nalewaja. All-instances oblivious chase termination is undecidable for single-head binary tgds. IJCAI, 2020.

[24] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, September 1984.

[25] Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura. Benchmarking the chase. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 37–52, 2017.

[26] Meghyn Bienvenu, Pierre Bourhis, Marie-Laure Mugnier, Sophie Tison, and Federico Ulliana. Ontology-mediated query answering for key-value stores. In *IJCAI: International Joint Conference on Artificial Intelligence*, 2017.

[27] Elena Botoeva, Diego Calvanese, Benjamin Cogrel, Martin Rezk, and Guohui Xiao. OBDA Beyond Relational DBs: A Study for MongoDB. In *Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22-25, 2016.*, 2016.

[28] Pierre Bourhis, Michel Leclère, Marie-Laure Mugnier, Sophie Tison, Federico Ulliana, and Lily Gallois. Oblivious and semi-oblivious boundedness for existential rules. In *IJCAI 2019-International Joint Conference on Artificial Intelligence*, 2019.

[29] Flavien Boussuge, Christopher M Tierney, Harold Vilmart, Trevor T Robinson, Cecil G Armstrong, Declan C Nolan, Jean-Claude Léon, and Federico Ulliana. Capturing simulation intent in an ontology: Cad and cae integration application. *Journal of Engineering Design*, 30(10-12):688–725, 2019.

[30] Patrice Buche, Juliette Dibie-Barthelemy, Liliana Ibanescu, and Lydie Soler. Fuzzy web data tables integration guided by an ontological and terminological resource. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):805–819, 2011.

[31] Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2):146–193, 2000.

[32] Damian Bursztyn, François Goasdoué, and Ioana Manolescu. Teaching an RDBMS about ontological constraints. *Proceedings of the VLDB Endowment*, 2016.

[33] Marco Calautti, Georg Gottlob, and Andreas Pieris. Chase termination for guarded existential rules. In Tova Milo and Diego Calvanese, editors, *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 91–103. ACM, 2015.

[34] Marco Calautti and Andreas Pieris. Oblivious chase termination: The sticky case. In *22nd International Conference on Database Theory, ICDT 2019 (to appear)*, 2019.

[35] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, KR'08, page 70–80. AAAI Press, 2008.

[36] A. Calì, G. Gottlob, and T. Lukasiewicz. A General Datalog-Based Framework for Tractable Query Answering over Ontologies. In *PODS'09*, pages 77–86, 2009.

[37] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

[38] Andrea Calı, Georg Gottlob, and Andreas Pieris. Query rewriting under non-guarded rules. *Proc. AMW*, 2010.

[39] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering sparql queries over relational databases. *Semantic Web*, 8(3):471–487, 2017.

[40] 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.

[41] 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.

[42] Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Verification of evolving graph-structured data under expressive path constraints. In *19th International Conference on Database Theory*, 2016.

[43] David Carral, Irina Dragoste, Larry González, Ceriel Jacobs, Markus Krötzsch, and Jacopo Urbani. Vlog: A rule engine for knowledge graphs. In *International Semantic Web Conference*, pages 19–35. Springer, 2019.

[44] David Carral, Irina Dragoste, and Markus Krötzsch. Detecting chase (non)termination for existential rules with disjunctions. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 922–928. ijcai.org, 2017.

[45] David Carral, Irina Dragoste, Markus Krötzsch, and Christian Lewe. Chasing sets: How to use existential rules for expressive reasoning. In *IJCAI*, 2019.

[46] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, WWW Alt. '04, page 74–83, New York, NY, USA, 2004. Association for Computing Machinery.

[47] A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. Comput.*, 14(3):671–677, 1985.

[48] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90, 1977.

[49] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.

[50] Peinan Chen, Vijay Gadepally, and Michael Stonebraker. The BigDAWG monitoring framework. In *High Performance Extreme Computing Conference (HPEC), 2016 IEEE*, pages 1–6. IEEE, 2016.

[51] Hubert Comon, Max Dauchet, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree automata techniques and applications*. 1997.

[52] Olivier Corby and Catherine Faron Zucker. Corese: A corporate semantic web engine. In *International Workshop on Real World RDF and Semantic Web Applications, International World Wide Web Conference*, 2002.

[53] Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *ACM Symposium on Theory of Computing*, pages 477–490, 1988.

[54] Stathis Delivorias. *Chase Variants & Boundedness. (Caractérisation des Bornes de Chaînage Avant en Règles Existentielles (Datalog+))*. PhD thesis, University of Montpellier, France, 2019.

[55] Stathis Delivorias, Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. On the k-boundedness for existential rules. In *International Joint Conference on Rules and Reasoning*, pages 48–64. Springer, 2018.

[56] Stathis Delivorias, Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. Characterizing boundedness in chase variants. *Theory Pract. Log. Program.*, 21(1):51–79, 2021.

[57] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In Maurizio Lenzerini and Domenico Lembo, editors, *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. ACM, 2008.

[58] Ronald Fagin. A normal form for relational databases that is based on domians and keys. *ACM Trans. Database Syst.*, 6(3):387–415, 1981.

[59] Ronald Fagin, Phokion G Kolaitis, Renée J Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[60] 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.

[61] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of

*Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer, June 2012.

[62] Vijay Gadepally, Jeremy Kepner, and Albert Reuther. Storage and database management for big data. In *Big Data: Storage, Sharing, and Security*, pages 15–41. Auerbach Publications, 2016.

[63] Lily Gallois. *Dialogue entre la procédure du chase et la réécriture sur les mots. (Dialog between chase approach and string rewriting system approach)*. PhD thesis, University of Lille, France, 2019.

[64] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. That's all folks! llunatic goes open source. *Proc. VLDB Endow.*, 7(13):1565–1568, August 2014.

[65] Tomasz Gogacz and Jerzy Marcinkowski. All–instances termination of chase is undecidable. In *International Colloquium on Automata, Languages, and Programming*, pages 293–304. Springer, 2014.

[66] Tomasz Gogacz, Jerzy Marcinkowski, and Andreas Pieris. All-instances restricted chase termination. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 245–258, 2020.

[67] Roy Goldman and Jennifer Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 436–445, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[68] Georg Gottlob. Knowledge processing, logic, and the future of ai. Vienna World Logic Day Lecture, 2021.

[69] Georg Gottlob and Christoph Koch. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM*, 10, 2004.

[70] Gösta Grahne and Adrian Onet. Anatomy of the chase. *Fundam. Inform.*, 157(3):221–270, 2018.

[71] 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.*, 47:741–808, 2013.

[72] Jiaqi Gu, Yugo H. Watanabe, William A. Mazza, Alexander Shkapsky, Mohan Yang, Ling Ding, and Carlo Zaniolo. Rasql: Greater power and performance for big data analytics with recursive-aggregate-sql on spark. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 467–484, New York, NY, USA, 2019. Association for Computing Machinery.

[73] L Hallau, M Neumann, B Klatt, B Kleinhenz, T Klein, C Kuhn, M Röhrig, C Bauckhage, K Kersting, A-K Mahlein, et al. Automated identification of sugar beet diseases using smartphones. *Plant pathology*, 67(2):399–410, 2018.

[74] André Hernich. Computing universal models under guarded tgds. In Alin Deutsch, editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 222–235. ACM, 2012.

[75] André Hernich and Nicole Schweikardt. Cwa-solutions for data exchange settings with target dependencies. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 113–122, 2007.

[76] Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. Undecidable boundedness problems for datalog programs. *Journal of Logic Programming*, 25(2):163–190, 1995.

[77] Liliana Ibanescu, Juliette Dibie, Stéphane Dervaux, Elisabeth Guichard, and Joe Raad. Po2-a process and observation ontology in food science. application to dairy gels. In *Research Conference on Metadata and Semantics Research*, pages 155–165. Springer, 2016.

[78] Yannis E. Ioannidis. A time bound on the materialization of some recursively defined views. In *Proceedings of the 11th International Conference on Very Large Data Bases - Volume 11*, VLDB '85, page 219–226. VLDB Endowment, 1985.

[79] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[80] Evgeny Kharlamov, Dag Hovland, Martin G. Skjæveland, Dimitris Bilidas, Ernesto Jiménez-Ruiz, Guohui Xiao, Ahmet Soylu, Davide Lanti, Martin Rezk, Dmitriy Zheleznyakov, Martin Giese, Hallstein Lie, Yannis E. Ioannidis, Yannis Kotidis, Manolis Koubarakis, and Arild Waaler. Ontology based data access in statoil. *J. Web Sem.*, 44:3–36, 2017.

[81] Mickael Kifer. Rules and Ontologies in F-logic. In *RW*, 2005.

[82] Mickael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 1995.

[83] Mélanie König, Michel Leclere, and Marie-Laure Mugnier. Query rewriting for existential rules with compiled preorder. In *IJCAI: International Joint Conference on Artificial Intelligence*, pages 3006–3112, 2015.

[84] 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.

[85] George Konstantinidis and José Luis Ambite. Optimizing the chase: Scalable data integration under constraints. *Proc. VLDB Endow.*, 7(14):1869–1880, 2014.

[86] Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Veronika Thost. Attributed description logics: Reasoning on knowledge graphs. In *IJCAI*, pages 5309–5313, 2018.

[87] Markus Krötzsch, Maximilian Marx, and Sebastian Rudolph. The power of the terminating chase (invited talk). In *22nd International Conference on Database Theory (ICDT 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[88] Michel Leclère, Marie-Laure Mugnier, Michaël Thomazo, and Federico Ulliana. A single approach to decide chase termination on linear existential rules. In *ICDT 2019-International Conference on Database Theory*, 2019.

[89] Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. On bounded positive existential

rules. In Maurizio Lenzerini and Rafael Peñaloza, editors, *Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22-25, 2016.*, volume 1577 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[90] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 233–246, 2002.

[91] Maurizio Lenzerini. Managing data through the lens of an ontology. *AI Magazine*, 39(2):65–74, 2018.

[92] Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi. Metamodeling and metaquerying in OWL2QL. *Artificial Intelligence*, 292:103432, 2021.

[93] Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic el using a relational database system. In *IJCAI*, volume 9, pages 2070–2075, 2009.

[94] David Maier, Jeffrey D Ullman, and Moshe Y Vardi. On the foundations of the universal relation model. *ACM Transactions on Database Systems (TODS)*, 9(2):283–308, 1984.

[95] Jerzy Marcinkowski. Achilles, turtle, and undecidable boundedness problems for small datalog programs. *Society for Industrial and Applied Mathematics Journal on Computing*, 29(1):231–257, 1999.

[96] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In Jan Paredaens and Jianwen Su, editors, *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. ACM, 2009.

[97] Marie-Laure Mugnier. Data access with horn ontologies: Where description logics meet existential rules. *Künstliche Intell.*, 34(4):475–489, 2020.

[98] Marie-Laure Mugnier, Marie-Christine Rousset, and Federico Ulliana. Ontology-mediated queries for nosql databases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[99] Jeffrey F Naughton. Data independent recursion in deductive databases. *Journal of Computer and System Sciences*, 38(2):259–289, 1989.

[100] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. Rdfox: A highly-scalable rdf store. In *International Semantic Web Conference*, pages 3–20. Springer, 2015.

[101] Adrian Onet. *The chase procedure and its applications*. PhD thesis, Concordia University, Canada, 2012.

[102] Piotr Ostropolski-Nalewaja, Jerzy Marcinkowski, David Carral, and Sebastian Rudolph. A journey to the frontiers of query rewritability. *CoRR*, abs/2012.11269, 2020.

[103] M Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems, 4th Edition*, volume 2. Springer, 2020.

[104] Olivier Palombi, Federico Ulliana, Valentin Favier, Jean-Claude Léon, and Marie-Christine Rousset. My corporis fabrica: an ontology-based tool for reasoning and querying on complex

anatomical models. *Journal of biomedical semantics*, 5(1):20, 2014.

[105] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semant.*, 10:133–173, 2008.

[106] Emil L. Post. Recursive unsolvability of a problem of thue. In *Journal of Symbolic Logic*, page 1–11, 1947.

[107] Pierre-Yves Rabattu, Benoit Massé, Federico Ulliana, Marie-Christine Rousset, Damien Rohmer, Jean-Claude Léon, and Olivier Palombi. My corporis fabrica embryo: An ontology-based 3d spatio-temporal modeling of human embryo development. *Journal of biomedical semantics*, 6(1):1–15, 2015.

[108] Swan Rocher. *Querying Existential Rule Knowledge Bases: Decidability and Complexity. (Interrogation de Bases de Connaissances avec Règles Existentielles : Décidabilité et Complexité)*. PhD thesis, University of Montpellier, France, 2016.

[109] Olivier Rodriguez, Reza Akbarinia, and Federico Ulliana. Querying key-value stores under single-key constraints: Rewriting and parallelization. In *International Joint Conference on Rules and Reasoning*, pages 198–206. Springer, 2019.

[110] Olivier Rodriguez, Marie-Laure Mugnier, and Federico Ulliana. Tree rewriting under constraints (technical report). 2021.

[111] Cornelius Rosse, José L Mejino, Bharath R Modayur, Rex Jakobovits, Kevin P Hinshaw, and James F Brinkley. Motivation and organizational principles for anatomical knowledge representation: the digital anatomist symbolic knowledge base. *Journal of the American Medical Informatics Association*, 5(1):17–40, 1998.

[112] Marie-Christine Rousset and Federico Ulliana. Extracting bounded-level modules from deductive rdf triplestores. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[113] Sebastian Rudolph and Markus Krötzsch. Flag & check: Data access with monadically defined queries. In *Proc. 32nd Symposium on Principles of Database Systems (PODS'13)*, pages 151–162. ACM, June 2013.

[114] Yehoshua Sagiv. On computing restricted projections of representative instances. In *Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '85, page 171–180, New York, NY, USA, 1985. Association for Computing Machinery.

[115] Pierre Senellart. Provenance and probabilities in relational databases. *SIGMOD Rec.*, 46(4):5–15, 2017.

[116] Michael Stonebraker. ACM SIGMOD blog: The case for polystores, 2015.

[117] M. Thomazo, J.-F. Baget, M.-L. Mugnier, and S. Rudolph. A Generic Querying Algorithm for Greedy Sets of Existential Rules. In *KR*, 2012.

[118] Michaël Thomazo. *Conjunctive Query Answering Under Existential Rules - Decidability, Complexity, and Algorithms*. PhD thesis, Montpellier 2 University, France, 2013.

[119] Michaël Thomazo, Jean-François Baget, Marie-Laure Mugnier, and Sebastian Rudolph. A

generic querying algorithm for greedy sets of existential rules. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, 2012.

[120] Efthymia Tsamoura, David Carral, Enrico Malizia, and Jacopo Urbani. Materializing knowledge bases via trigger graphs. *Proceedings of the VLDB Endowment*, 2021.

[121] Federico Ulliana, Jean-Claude Léon, Olivier Palombi, Marie-Christine Rousset, and François Faure. Combining 3d models and functions through ontologies to describe man-made products and virtual humans: Toward a common framework. *Computer-Aided Design and Applications*, 12(2):166–180, 2015.

[122] Harold Vilmart, Jean-Claude Léon, and Federico Ulliana. Extraction et inférence de connaissances à partir d'assemblages mécaniques définis par une représentation cao 3d. In *EGC: Extraction et Gestion des Connaissances*, pages 21–32, 2017.

[123] Harold Vilmart, Jean-Claude Léon, and Federico Ulliana. From cad assemblies toward knowledge-based assemblies using an intrinsic knowledge-based assembly model. *Computer-Aided Design and Applications*, 15(3):300–317, 2018.

[124] Zhe Wang, Kewen Wang, and Xiaowang Zhang. Forgetting and unfolding for existential rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[125] Gio Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, March 1992.

[126] Gio Wiederhold. Intelligent integration of information. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 434–437, 1993.