



Chase Variants & Boundedness

Stathis Delivorias

► To cite this version:

Stathis Delivorias. Chase Variants & Boundedness. Logic in Computer Science [cs.LO]. Université de Montpellier; LIRMM, University of Montpellier; INRIA Sophia Antipolis - Méditerranée, 2019. English. NNT : . tel-02408597v1

HAL Id: tel-02408597

<https://theses.hal.science/tel-02408597v1>

Submitted on 13 Dec 2019 (v1), last revised 12 Feb 2020 (v2)

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.

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En INFORMATIQUE

École doctorale I2S

Unité de recherche LIRMM

Chase Variants & Boundedness

Présentée par Stathis DELIVORIAS
Le 26 Septembre 2019

Sous la direction de Marie-Laure MUGNIER
et Federico ULLIANA
et Michel LECLÈRE

Devant le jury composé de

David DELAHAYE, Professeur, Université de Montpellier

Sophie TISON, Professeure, Université de Lille

Chan LE DUC, Maître de Conférences, Université Paris 8

Sebastian RUDOLPH, Professeur, Université Polytechnique de Dresde, Allemagne

Meghyn BIENVENU, Chercheuse CNRS, Université de Bordeaux

Marie-Laure MUGNIER, Professeure, Université de Montpellier

Federico ULLIANA, Maître de Conférences, Université de Montpellier

Michel LECLÈRE, Maître de Conférences, Université de Montpellier

Examineur - Président du jury

Rapporteuse

Rapporteur

Examineur

Examinatrice

Directrice de thèse

Co-encadrant de thèse

Co-encadrant de thèse



UNIVERSITÉ
DE MONTPELLIER

Abstract

The chase is a family of algorithms designed to infer data with the use of ontological knowledge, which is encoded in existential rules, a sub-language of first-order logic. A considerable literature has been devoted to its analysis, approaching it from a variety of presupposed terminological and notational background. We define a unifying framework for the specification and study of chase algorithms. We utilize it to compare and clarify the properties that discern the different variants of the chase. We particularly focus on studying whether there is a bound to the maximum length of a chain of interdependent rule applications (where interdependency means that the output of a rule application is contributing to triggering the next rule application). This is the problem of boundedness, or k -boundedness, when the bound k is given. By investigating a number of intermediate properties, we find that k -boundedness is decidable for several chase variants. In addition to other secondary results, we define two new chase variants with the aim of reducing redundant rule applications without heavily increasing the computation cost.

Résumé

Le « chase » est une famille d'algorithmes conçus pour inférer des données en utilisant des connaissances ontologiques représentées par des règles existentielles, un sous-langage de la logique du premier ordre. Une littérature importante concerne son analyse, mais utilise des notations et des terminologies variées. On définit un cadre unificateur pour la spécification et l'étude des algorithmes du chase. On utilise ce cadre pour expliciter et comparer les propriétés des différentes variantes du chase. On se focalise particulièrement sur le problème de la " k -saturation-bornée" : k est-elle la taille maximum d'une chaîne d'applications de règles interdépendantes (où interdépendance signifie que le résultat d'une application d'une règle contribue au déclenchement de l'application suivante) ? En définissant des propriétés intermédiaires, on montre que le problème de la k -saturation-bornée est décidable pour de nombreuses variantes du chase. Parmi d'autres résultats, nous définissons deux nouvelles variantes du chase qui réduisent le nombre d'applications de règles redondantes sans augmenter significativement le temps de calcul.

Acknowledgements

This work is dedicated to my mother and to my father, for their love, support, insight and honesty.

I would like to thank my supervisors Marie-Laure, Federico & Michel, for their guidance and patience and for giving me a chance to work in this challenging scientific field in excellent conditions. During these years I have learnt much more than what I imagined that there is to learn. I am grateful for the people I have met, teachers, colleagues and friends, in Montpellier as well as earlier in my life. A special thanks goes to Morgane, for her care and support amid periods of heavy workload.

Boundedness in existential rules is an intrinsically complex subject. I hope that this thesis contributes to clarifying a number of notions and inspires some readers towards the advancement of this research.

Contents

1	Introduction	3
2	Preliminaries	12
2.1	The Language of Existential Rules	12
2.1.1	Atomsets & Rulesets	12
2.1.2	Substitutions & Logical Entailment	14
2.1.3	Redundancy & Retraction	16
2.2	Query Answering over Knowledge Bases	20
2.2.1	Knowledge Base & Universal Model	20
2.2.2	Forward Chaining	22
2.2.3	Backward Chaining	27
3	Boundedness in the Literature	29
3.1	Datalog	29
3.2	Existential Rules	33
4	The Chase	36
4.1	Derivations	36
4.2	Monotonic Chase Variants	47
4.3	Non-monotonic Chase Variants	54
4.3.1	Vacuum Chase & Frugal Chase	55
4.3.2	Parallel Chase & Core Chase	59
4.3.3	Submonotonicity	64
4.4	Comparing Chase Variants	65
4.4.1	Termination & Elimination of Redundancy	65

4.4.2	The Breadth-first Approach	69
4.5	Chase Graphs & Chase Space	83
4.5.1	Ancestors & Descendants	84
4.5.2	Chase Graphs	86
4.5.3	The Chase Space	93
5	Characterizing Boundedness in Different Chase Variants	104
5.1	Boundedness & k -Boundedness	104
5.1.1	Boundedness	105
5.1.2	k -Boundedness	107
5.2	Preservation of Ancestry	109
5.2.1	The Link with k -Boundedness	110
5.2.2	Heredity	113
5.2.3	bf- R -Compliance	117
5.2.4	V-, F-, E- and C-Chase Do Not Preserve Ancestry . .	133
5.3	Complexity Upper Bounds	136
5.4	Towards The Limits of bf- R -Compliance	141
5.4.1	Local Core Chase	142
5.4.2	The Conjecture	146
5.5	k -Minimal Chase Graphs	149
5.5.1	A Characterization of k -Boundedness	149
5.5.2	Computing k -Minimal X-Chase Graphs	153
6	Conclusion	159
6.1	Summary	159
6.2	Future Work	161
	APPENDIX	166
(A)	Query Rewriting in Existential Rules	166
(B)	Properties of the Semi-Oblivious Chase	169
(C)	Partial Core & Freezing	174
(D)	Frugal Chase	176

1 Introduction

In an era where an abundance of data is available, the development of formalisms to represent and reason with *ontological knowledge* (i.e. general/abstract information on an application domain) can aid in the management and in the utilization of this data. *Existential rules* [1, 2] are a positive fragment of first-order logic that is used to represent ontological knowledge. Existential rules are of the form “if *body* then *head*”, where the body and the head are conjunctions of atomic formulas. As an example take the rule $\forall x (human(x) \rightarrow \exists y human(y) \wedge motherOf(y, x))$ which states that every human x has a human mother y . In the following, to simplify notation we will not mention the universal quantifier “ \forall ” because it is implicit that every variable that appears in the body of a rule is universally quantified.

Existential rules extend the deductive database language known as *Datalog* [3], which is why they are also known as *Datalog*⁺. In Datalog, all the variables that are in the rule head necessarily appear also in the body. Hence those rules cannot infer the existence of new individuals. A simple example of such a rule is $sibling(x, y) \rightarrow sibling(y, x)$. However, in an open domain perspective, it cannot be assumed that all the relevant individuals are known in advance. That is why the ability of asserting the existence of unknown individuals has been recognized as a desired feature of ontological languages. Such a feature is offered by knowledge representation languages like description logics (even the simplest ones as *DL-Lite* [4] and \mathcal{EL} [5]) as well as existential rules.

Existential rules were initially studied under the name *tuple-generating dependencies* as database constraints [6], but in the recent years they have attracted interest as an ontological language, mainly used for *ontology-mediated*

query-answering (see e.g. the survey chapters [7, 8]). In this context a *knowledge base* comprises a set of existential rules (sometimes also called *ontology* since it is expressing general domain knowledge) and a *factbase* which is an existentially closed conjunction of atomic formulas (also called *atoms*) and serves as a logical abstraction of a database. Given a knowledge base like

$$\begin{aligned} & \textit{sibling}(x, y) \rightarrow \textit{sibling}(y, x) \\ & \textit{sibling}(\textit{August}, \textit{May}) \end{aligned}$$

we are interested to know the answer to a *query* like $\textit{sibling}(\textit{May}, z)$, which asks for all values of z such that $\textit{sibling}(\textit{May}, z)$ is true for the knowledge base. Here, $z = \textit{August}$ is an answer (notice how the rule is needed in order to infer this answer). More generally, most work in the area considers conjunctive queries, which are existentially quantified conjunctions of atoms. The free variables that might occur in a query are the answer variables. Consequently, the (*conjunctive*) *query answering* problem asks, given a knowledge base and a conjunctive query, whether there is a substitution of the free variables of the query by constants such that the knowledge base entails the substituted query.

In this thesis we will be interested in a decision problem which is polynomially equivalent with the (conjunctive) query answering problem [1]: that is the *Boolean conjunctive query (BCQ) answering* problem, where the conjunctive query does not have any free variable (so it is an existentially closed conjunction) and the answer is yes or no, depending on whether the query is entailed by the knowledge base. Two examples of BCQs on the previous knowledge base are $\textit{sibling}(\textit{May}, \textit{August})$ and $\exists w \exists z \textit{sibling}(w, z) \wedge \textit{sibling}(z, w)$. Both those queries are entailed by the knowledge base. However the query $\exists z \textit{sibling}(z, z)$ is not entailed.

One of the standard approaches to solving the BCQ problem is known as *materialization* or *forward chaining* (see e.g. [9, 10]). In this approach, we use the rules to infer more knowledge, expanding the factbase. Then the query can be directly evaluated with respect to the new factbase. The drawback here is that forward chaining does not always terminate: this depends on the knowledge base. Indeed it is known that the BCQ answering problem is undecidable for the general language of existential rules (from [11] on tuple-generating

dependencies). As a result, the quest for sub-languages (usually specified by imposing restrictions on the form of the rules) for which the forward chaining halts has been a prevailing objective in the research community. As an example, take the knowledge base

$$\begin{aligned} human(x) &\rightarrow \exists y \, human(y) \wedge motherOf(y, x) \\ human(April) \end{aligned}$$

By applying the rule on the only atom of our factbase, we entail the expanded factbase $\exists y_0 (human(y_0) \wedge motherOf(y_0, April) \wedge human(April))$. Here y_0 is a variable that is introduced to our factbase to represent the “mother of” April. Now our rule is applicable on $human(y_0)$, and the application will produce a new individual (variable) y_1 , similarly connected to y_0 as y_0 is connected to April, i.e. y_1 will serve as the “mother of” y_0 . In this way we can create a factbase of unlimited size, representing a chain of ancestors of April. It is evident that in this case the forward chaining does not terminate.

The predominant feature of existential rules is their ability to refer to the existence of new individuals that fulfill particular properties. But using rules in the above way to introduce variables to the factbase brings about a certain inconvenience: the added atoms might not really express new knowledge. It might be the case that the produced factbase is logically equivalent with the initial one. If for example we start from the knowledge base

$$\begin{aligned} parentOf(x, y) &\rightarrow \exists z \, parentOf(z, y) \wedge haveCommonChild(x, z) \\ haveCommonChild(x, y) &\rightarrow haveCommonChild(y, x) \\ parentOf(Venus, April) \end{aligned}$$

then the first rule can be applied, which then allows an application of the second rule. This first “round” of rule applications does provide new information as it asserts that there exists a new individual z_0 that has had a child (April) with Venus. Indeed the resulting factbase is

$$\begin{aligned} \exists z_0 (parentOf(Venus, April) \wedge parentOf(z_0, April) \wedge \\ \wedge haveCommonChild(Venus, z_0) \wedge haveCommonChild(z_0, Venus) \end{aligned}$$

But if we reapply the first rule on the new atom $\text{parentOf}(z_0, \text{April})$, we will produce a new (unknown) individual z_1 which will be *redundant* since it will have exactly the same properties as Venus. In this way we can continue to apply the rules to new atoms, without actually producing new information. This motivates the definition of algorithms that will regulate rule applications with the aim of avoiding the addition of useless atoms in the factbase.

The algorithms used to perform forward chaining are collectively known as *the chase* [12, 13, 14]. Many chase algorithms have been defined, such as the *oblivious chase* [15], the *semi-oblivious chase* (also known as *skolem chase*) [16], the *restricted chase* (also known as the *standard chase*) [17] and the *core chase* [18]. Each variant of the chase imposes restrictions on the choice of rule applications to be made and the subsequent evolution of the factbase. Most of the chase variants (and all the chase variants that are presented in this thesis) produce logically equivalent results which moreover represent a universal model of the knowledge base, i.e. a model that can be mapped by homomorphism to any other model of the knowledge base. This universal model property is key because it allows one to recast the BCQ problem as a classical query evaluation problem on the factbase produced by the chase (provided that the considered chase halts). Actually, every chase variant halts under different circumstances, hence the question rises of whether we can predetermine, based on some syntactic conditions, whether a particular chase variant will terminate on a given knowledge base. Unfortunately this has been shown to be undecidable for all major chase variants [16, 18]. As a consequence there has been interest in finding properties in sets of rules that guarantee that the chase terminates for every factbase. Even though again, the general problem is undecidable [1, 19], there is much literature devoted to finding sufficient conditions for *chase termination* (also known as *all-instance termination*) of particular sub-languages of existential rules [1, 20, 21, 22, 23].

A decision problem that relates to chase termination is that of *boundedness*: A set of rules is *bounded* if there is a bound to the *depth* of the chase independently of the factbase. This depth corresponds to the maximal depth of a proof

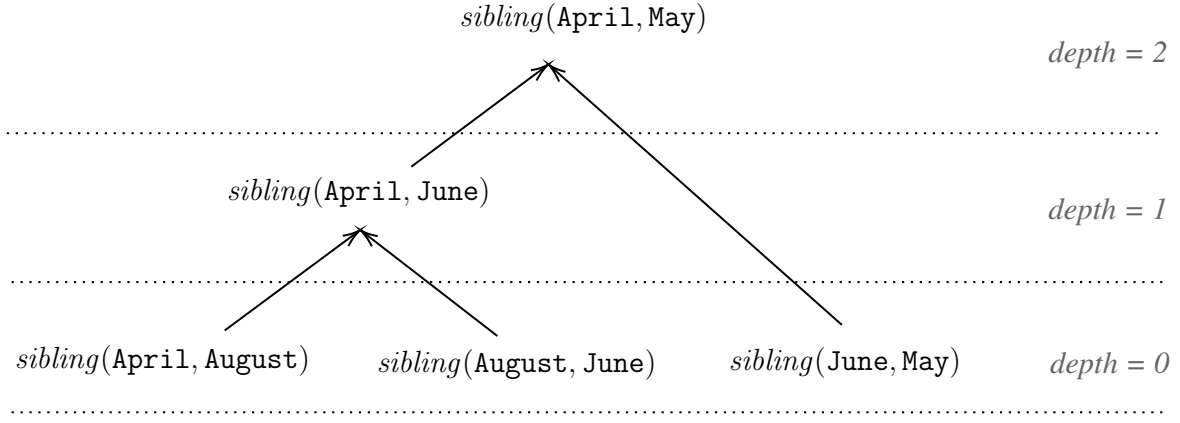


Figure 1.1: A *chase graph*, illustrating the concept of the depth of the chase.

of an inferred fact. As an example consider the single transitive rule

$$sibling(x, y) \wedge sibling(y, z) \rightarrow sibling(x, z)$$

and notice that even though this is a datalog rule, i.e. it does not introduce new variables and the chase is necessarily terminating for every factbase, it does not constitute a bounded (set of) rule(s): if the initial factbase is $sibling(\text{April}, \text{August}) \wedge sibling(\text{August}, \text{June})$ then the only atom that can be inferred is $sibling(\text{April}, \text{June})$, and it will be inferred with one rule application directly from the initial factbase, hence the depth of the chase will be 1. However, starting with the factbase

$$sibling(\text{April}, \text{August}) \wedge sibling(\text{August}, \text{June}) \wedge sibling(\text{June}, \text{May})$$

we can also infer the atom $sibling(\text{April}, \text{May})$ by using our rule, the atom $sibling(\text{June}, \text{May})$ which is in the initial factbase and the atom $sibling(\text{April}, \text{June})$ which is inferred at depth 1. In this case the depth of the chase is 2 (see Figure 1.1). By extending the initial factbase in this manner, we see that there is no bound to the depth of the chase with a transitive rule, even if the chase always terminates (i.e. chase termination does not imply boundedness). On the other hand, boundedness ensures chase termination, as well as several other semantic properties like the *first-order rewritability* property [24] (also known as *finite unification set* property [1]) which states that the

query can be rewritten to a first order formula which is entailed by the factbase if and only if the original query is entailed by the knowledge base. This implies that many interesting static analysis problems such as query containment under existential rules are decidable when a set of rules is bounded. Finally, as noted in some of the first publications that studied boundedness in Datalog [6, 25, 26], boundedness can be used to measure the extent of recursivity of a ruleset, i.e. the maximal number of times that we would need to reapply the same cycle of rules before the chase terminates.

Our Contribution

This dissertation focuses on two major subjects that complement each other:

1 Chase Variants

We define a unifying framework which permits the modelling of all major known chase variants for existential rules. Using a refined definition of the concept of *derivation* which encodes with precision a sequence of rule applications and their effect on the factbase, we are able to formally specify the notion of *chase variant* as a class of derivations. This allows the definition of properties which facilitate the comparison of chase variants and can be used to obtain concrete technical results. So while derivations allow us to model forward chaining in fine detail, we are also able to abstract away when discussing properties of some hypothetical chase variant X (also called *X-chase*).

It is known that factbases can be represented as graphs/hypergraphs, and that graph homomorphisms then correspond to logical entailment [27, 1]. In this dissertation we exploit the graph-theoretical view of existential rules, by using exclusively (hyper)graph theory (and elementary set theory) in all proofs and technical parts. More importantly, we accentuate the connection between *redundancy* in the factbase and the existence of a *retraction*, which is a specific type of graph homomorphism, from the factbase to a sub-factbase. Consequently, we bring to light the link between chase variants and retractions, since the main focus of chase variants is the elimination of redundancy which

is caused by the introduction of new variables to the factbase by the existential rules. Note that in Datalog this discussion trivially disappears along with the existential variables because there is no redundancy in factbases that do not include any variables. However, redundancy in the ruleset remains pertinent in Datalog but our line of research does not touch on this matter.

The new definition of chase variant trivially facilitates the specification of “new” chase variants: it suffices to use any random criterion that restricts the considered derivations to a specific class. Under this definition, a restriction on the order of rule applications (e.g. the breadth-first approach) or even a restriction on the rule classes considered (e.g. acyclic rules) can constitute a different chase variant. This is consistent, since these restrictions can greatly affect the overall behavior of the chase variant (with respect to termination, boundedness, etc).

But in this thesis we have also contributed two “really new” chase variants, the *vacuum chase* and the *local core chase*. Both optimize in different ways the existing chase variants with respect to eliminating redundancy. The first is an optimization of the *frugal chase* [28], while the second is an intermediate algorithm between the breadth-first restricted chase and the core chase. The local core chase also seems to behave well with respect to boundedness, although our time limitations only allowed us to include a conjecture with regard to that matter in this work.

2 Boundedness

Boundedness has been largely studied in Datalog [29, 30, 31, 25], where it has been shown to be an undecidable property in the general case. However, in the domain of existential rules there has been little related work. The first step is to recognize that boundedness has to be parametrized by the respective chase variant X . This is because, if X and Y are two different chase variants, a set of rules might be *X-bounded* (i.e. bounded with respect to the X -chase) but not Y -bounded. Then, since we already knew that the question

“given a set of rules, is there a bound k to the depth of the X -chase?”

is undecidable unless we restrict the rule language (because of its undecidability in Datalog), we pose the question

“given a set of rules, is the number k a bound to the depth of the X-chase?”

This is the problem of *X- k -boundedness*, where the bound k is given and part of the question. Notice that in both cases, we are researching the depth of the X-chase with a given set of rules and any factbase. In the informal explanatory parts of the thesis, when we refer to the properties in general and not concerning a particular X-chase, we still keep the names *boundedness* and *k-boundedness* instead of X-boundedness and X- k -boundedness respectively.

A *chase graph* is a partial representation of a derivation where the nodes are all the atoms that appear in the derivation and the edges indicate which atoms are used in order to produce other atoms. Since boundedness relates to the depth of the chase, the notion of chase graph is central to obtaining an intuition of the mechanisms which influence whether a set of rules is bounded or not. We have largely exploited this perspective. At first we specified a property called *preservation of ancestry* which can be understood as the invariance of a part of the chase graph when we reduce the initial factbase. We showed that if a chase variant X preserves ancestry, then X- k -boundedness is decidable. Then we also used some other intermediate properties to finally conclude that many chase variants (such as the oblivious, the semi-oblivious, the restricted chase and their breadth-first versions) preserve ancestry. We therefore show that k -boundedness is decidable for many chase variants, while for all other chase variants we show that they do not preserve ancestry. This leaves open the question of boundedness for those chase variants. For the local core chase however, we hypothesize that a weaker property, named *loose preservation of ancestry* is satisfied. And loose preservation of ancestry ensures the decidability of k -boundedness.

Furthermore, we have also showed that in the case of *linear existential rules*, i.e. rules whose body consists of a single atom, when the X-chase preserves ancestry, it holds that X-chase termination is equivalent with X-boundedness. This solves the question of boundedness of linear sets of rules for a number

of chase variants (namely the oblivious, the semi-oblivious chase and their breadth-first versions), and shows a way towards answering the question for other chase variants. Note that here we are talking about boundedness and not just k -boundedness. Finally, we defined a certain kind of minimality of chase graphs which leads to a characterization of k -boundedness. This can contribute to solutions regarding the decidability of the problem of k -boundedness for the chase variants that do not (loosely) preserve ancestry.

Apart from the distinct contributions concerning boundedness and the definition of chase variants, this thesis includes a significant number of secondary results that relate to the chase. We have made an effort to address or at least mention as many as possible of the questions that rise when discussing and comparing those chase variants, especially whatever is related to depth. In addition, there are many examples and counter-examples that demonstrate the variety of different forward chaining scenarios.

2 Preliminaries

In this chapter we will formally introduce the main notions which we will be working on. The first part of the preliminaries is dedicated to the introduction of the elementary syntactic and semantic concepts on which our research is focused. Then we go on to present forward and backward chaining with existential rules. Therein we formulate the concept of boundedness. Throughout this thesis, we assume that the reader is familiar with the general concepts of graph theory as well as first-order logic [32].

2.1 The Language of Existential Rules

Our study concerns the sub-language of first-order logic called *Positive Existential Rules* or simply *Existential Rules*¹. This language has the very convenient feature that it can be completely encoded into graph theory. In this section we will present the basic components of existential rules and link them directly with the respective graph theoretic notions.

2.1.1 Atomsets & Rulesets

We work with first order formulas whose elementary syntactical entities are *predicates* (usually denoted with the letters p, q, r), *constants* (usually denoted with a, b, c) and *variables* (denoted with x, y, z). A *term* is a variable or a constant. Each predicate p is associated with a positive integer number, called the *arity* of p .

¹The term “positive” refers to the absence of negation/negative constraints.

The main syntactical element is an *atom*, which is of the form $p(t_1, \dots, t_n)$ where p is a predicate, n is the *arity* of p and t_1, \dots, t_n are terms. An *atomset* is a set of atoms. So given an atomset S , we will use the notation $\text{var}(S)$ to refer to the set variables that appear in S . The set of constants and terms that appear in the atoms of S are denoted with $\text{cnst}(S)$ and $\text{term}(S)$ respectively. In addition, we can use the same operators to refer to a single atom A , i.e. with $\text{var}(A)$, $\text{cnst}(A)$ and $\text{term}(A)$ we refer to the sets of variables, constants and terms of A respectively.

Definition 2.1. A *factbase* is the existential closure of a conjunction of atoms². We will usually symbolize a factbase with the letters F and Z . \dashv

Unless otherwise specified, we will always assume a factbase to be finite. It is very convenient to see factbases simply as atomsets. So for example $\{p(a, x), q(x, b, c)\}$ can represent the existentially closed conjunction $\exists x (p(a, x) \wedge q(x, b, c))$. Hence in this thesis, unless otherwise stated, by factbase we will refer to the corresponding atomset. Note that factbases like $\{p(a, x)\}$ and $\{p(a, y)\}$ are considered to be different syntactic entities.

Definition 2.2. An *existential rule* R is a first order formula of the form $\forall \bar{x} \forall \bar{y} [B(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}. H(\bar{x}, \bar{z})]$, where \bar{x} , \bar{y} and \bar{z} are disjoint sets of variables, and B and H are atomsets called the *body* and the *head* of the rule, respectively. The set of variables \bar{x} is shared by the body and the head of the rule, and it is called the *frontier* of the rule, denoted by $\text{fr}(R)$. The set \bar{z} is called the set of *existential variables* of the rule and it is denoted by $\text{exv}(R)$. The set $\bar{x} \cup \bar{y}$ is the set of *universally quantified variables* of R . The set $\bar{x} \cup \bar{y} \cup \bar{z}$ is called the set of *variables* of the rule and it is denoted by $\text{var}(R)$. The *disappearing variables* of an existential rule are those that are neither existential nor frontier, . We will call a finite set of existential rules (usually denoted by \mathcal{R}) a *ruleset*. \dashv

Henceforth, unless otherwise specified, *rule* means existential rule. In the literature, depending on the context, existential rules are also called *tuple generating dependencies* and a ruleset can be found to be referred to as an *ontology* or a

²i.e. a closed conjunction in prenex form, using only the existential quantifier.

program [11, 6, 33]. We will maintain the appellation ruleset, in order to avoid confusion. We will omit the universal quantifiers when representing existential rules. So for example instead of writing

$$\forall x \forall y \left(\left(p(x, y, x) \wedge q(y) \right) \rightarrow \exists z \left(r(z, x) \wedge q(z) \right) \right)$$

we will simply write $p(x, y, x) \wedge q(y) \rightarrow \exists z r(z, x) \wedge q(z)$.

In the following, it will be sometimes convenient to consider a rule R simply as a pair of atomsets (B, H) . Furthermore, we will use the notation $\text{body}(R) = B$ and $\text{head}(R) = H$. So the above rule can also be represented with $(\{p(x, y, x), q(y)\}, \{r(z, x), q(z)\})$. Of course, $\text{var}(\mathcal{R})$ denotes the set of variables that appear in a ruleset \mathcal{R} .

Existential Rules generalize the declarative logic programming language known as *Datalog* [3, 6]. In our framework, we say that a ruleset \mathcal{R} is said to be *datalog* if for all $R \in \mathcal{R}$ it holds that $\text{exv}(R) = \emptyset$. Another sub-language of existential rules which will interest us is that of *linear rules*. A ruleset \mathcal{R} is called *linear* if the head of every rule $R \in \mathcal{R}$ is composed of one single atom. Although in this thesis we will mainly be working with arbitrary rulesets, the particular classes of datalog rulesets and linear rulesets will be referenced in several instances.

2.1.2 Substitutions & Logical Entailment

In this subsection we present the basic terminology with regard to substituting terms with other terms in an atomset. The significance of this syntactic manipulation is that it has been shown to agree with the first-order logic semantics of the corresponding formulas. We will elaborate on this point below. In this dissertation we are employing the usual notation, found in most first order logic textbooks, for logical entailment (“ \models ”) and logical equivalence (“ \equiv ”).

A *substitution* σ is a mapping from a finite set of variables (its *domain*) to a set of terms with the condition that a variable in the domain cannot be mapped to itself. However we expand the notation by considering that σ acts as the

identity to any term that is not in its domain. Hence we can say that for every substitution σ and variable x it holds that $\sigma(x) = x$ if and only if $x \notin \text{dom}(\sigma)$, where $\text{dom}(\sigma)$ is the domain of σ . A substitution whose domain is empty is called the *identity substitution*. The set of σ -images of the domain variables $\{\sigma(x) \mid x \in \text{dom}(\sigma)\}$ is called the *codomain* of σ and we write $\text{codom}(\sigma)$.

We frequently denote a substitution as a set of mappings, for example $\{x \mapsto a, y \mapsto z\}$ is a substitution with domain $\{x, y\}$ and codomain $\{a, z\}$. Given two substitutions σ_1 and σ_2 , their *composition* $\sigma_1 \circ \sigma_2$ includes the union $\sigma_1 \cup \sigma_2$ and all the composed mappings $x \mapsto t$ where $x \neq t$ and there exists a variable y such that $x \mapsto y \in \sigma_1$ and $y \mapsto t \in \sigma_2$ (here t is a term).

Given an atomset F , we denote by $\sigma(F)$ the atomset obtained by substituting each variable in $\text{dom}(\sigma) \cap \text{var}(F)$ by its σ -image. In addition, the atomset $\sigma(F)$ is called a *specialization* of F . A *variable-renaming*, is a bijection between two sets of variables. Throughout this thesis, apart from greek lowercase letters like σ, π, τ, μ we will also be using classical function notation f, g and h for substitutions.

A *homomorphism* from an atomset F to an atomset F' is a substitution $h : \text{var}(F) \rightarrow \text{term}(F')$ such that $h(F) \subseteq F'$. It is known that a factbase F is logically entailed by a factbase F' , i.e. $F' \models F$ if and only if there exists a homomorphism from F to F' [6, 1]. This property is crucial to our methodology, in that we evade any calculus on traditional mathematical logic in order to attain our results.

Two atomsets F and F' are called *isomorphic* if there exists a variable renaming σ such that $\sigma(F) = F'$. In this case we call σ an *isomorphism* between F and F' . Notice that every isomorphism is a homomorphism whose inverse is also a homomorphism.

A substitution σ is a *unifier* of two atomsets F and F' with disjoint sets of variables³ if $\sigma(F) = \sigma(F')$. A unifier μ is a *most general unifier (mgu)* of F and F' if it is a unifier of F and F' , i.e. $\mu(F) = \mu(F')$, and for every other unifier σ of F and F' it holds that there exists a substitution θ such that $\sigma = \theta \circ \mu$ (where \circ represents the composition operator for two functions).

³ $\text{var}(F) \cap \text{var}(F') = \emptyset$.

Given a substitution σ and an atomset F , the *restriction* of σ on F , denoted with σ_F is the substitution which has as domain the set $\text{dom}(\sigma_F) = \text{dom}(\sigma) \cap \text{var}(F)$ where $\sigma_F(x) = \sigma(x)$ for every $x \in \text{dom}(\sigma_F)$.

In much of our work, we will be interested in properties of rulesets. Thus we will want to discuss how a ruleset behaves when interacting with any possible factbase. To this end, we will need a way to filter out some factbases from our search space. Consequently, we also define mappings between constants: Let F and F' be atomsets and $\tau : \text{cnst}(F) \rightarrow \text{cnst}(F')$, $\sigma : \text{var}(F) \rightarrow \text{var}(F')$ be mappings of constants and variables respectively. If $h = \tau \cup \sigma$ is a bijection and has the property that $h(F) = F'$ then h is a *quasi-isomorphism* from F to F' . Since quasi-isomorphism defines an equivalence relation on atomsets, given a set of factbases \mathcal{F} we call *quasi-equivalence classes* all equivalence classes with respect to quasi-isomorphism.

2.1.3 Redundancy & Retraction

In this subsection we settle the connection between atomsets and a particular type of hypergraphs (which can actually also be encoded in labelled directed graphs). This brings at our disposal an interesting array of handy notions and relevant results.

A *labelled ordered hypergraph* is a triple $H = (X, E, \mathcal{L})$, where E is a family of tuples of elements in X , i.e. a family of totally ordered subsets of X , and \mathcal{L} is a mapping from $X \cup E$ to a set of labels. We can consider an atomset F as a labelled ordered hypergraph $H = (X, E, \mathcal{L})$ where $X = \text{term}(F)$, E includes for every atom $p(t_1, \dots, t_n)$ in F the tuple (t_1, \dots, t_n) of its terms labelled by its predicate p , i.e. $\mathcal{L}((t_1, \dots, t_n)) = p$, and lastly

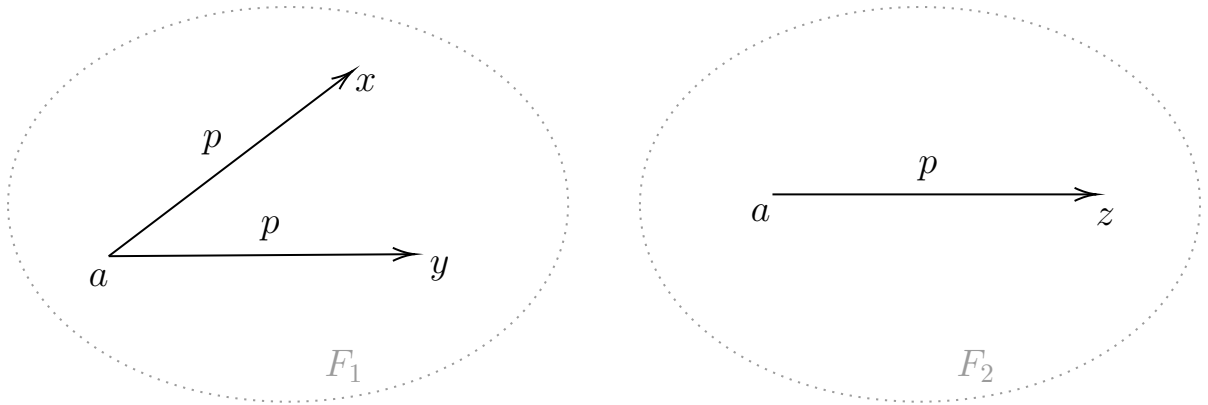
$$\mathcal{L}(t) = \begin{cases} \text{NULL} & t \in \text{var}(F) \\ t & t \in \text{cnst}(F) \end{cases}$$

and as established in [27, 1] there is a total correspondence between (labelled

ordered) hypergraph homomorphisms and atomset homomorphisms⁴. The label “NULL” assures that the actual naming of variables does not matter when considering atomset homomorphisms. As demonstrated in the following example, in figures we will drop this label for convenience and instead will mention the variable names.

Hypergraphs are useful as intuitive representations of factbases. In this regard, a rule $R = (B, H)$ can also be seen as a pair of hypergraphs (B and H). In this work we will be utilizing examples with factbases and rulesets of maximal arity 2 which correspond to traditional (labelled) graphs, hence are more easily illustrated.

Example 1: Let $F_1 = \{p(a, x), p(a, y)\}$ and $F_2 = \{p(a, z)\}$ (here a is a constant whereas x, y and z are variables). Here are the labelled graph representations of those factbases:



The substitution $\{x \mapsto z, y \mapsto z\}$ is a homomorphism from F_1 to F_2 , whereas the substitution $\{z \mapsto x\}$ is a homomorphism from F_2 to F_1 . Therefore the two factbases are logically equivalent. On the other hand there is no variable renaming that can act as a homomorphism between those two factbases, hence F_1 and F_2 are not isomorphic. ■

Notice that by applying the substitution $\{x \mapsto y\}$ we can reduce the factbase $\{p(a, x), p(a, y)\}$ to $\{p(a, y)\}$, which is a subset that is logically equivalent to the original factbase. This is an important operation and a central concept

⁴we assume the reader familiar with the notion of graph homomorphism. For this thesis the definition of factbase homomorphism given above suffices.

around which our work revolves. A substitution of this sort is called *retraction* and it is the main mechanism that can be used to eliminate redundancy from a factbase, i.e. removing one or several atoms from the factbase without changing the actual semantics of this factbase.

Definition 2.3. Let F be an atomset. A subset $F' \subseteq F$ is a *retract* of F if there exists a substitution $\sigma : (\text{var}(F) \setminus \text{var}(F')) \rightarrow \text{term}(F')$ with $\sigma(F) = F'$. In this case σ is said to be a *retraction* from F to F' (we also say that σ *acts as a retraction on F* or simply is a *retraction on F*). \dashv

Trivially, the identity substitution $\sigma : \emptyset \rightarrow \text{term}(F)$ is a retraction from F to F , and F is indeed a retract of F . We can say that F is the *trivial* retract of F (this will be needed when we want to specify that a retract is non-trivial). Throughout this thesis, the term *redundancy* of an atomset is used to refer to the existence of (non-trivial) retracts, which means that some of the variables are indeed redundant, since they represent information that is already expressed by other terms. When a variable is redundant, it can be identified with another term in the atomset. This means that all of the atoms where it appears are redundant as well.

Retractions are also known as *foldings* in the literature (although there can be minor differences in the definitions [34]). A retraction is a particular type of graph homomorphism (endomorphism). Retractions constitute an advantageous approach to the elimination of redundancy because they have some very good properties which facilitate their composition and decomposition.

Remark 2.1 (Composition of Retractions).

- i) Let g_1 be a retraction from F to F_1 and g_2 be a retraction from F_1 to F_2 . Then $g_2 \circ g_1$ is a retraction from F to F_2 .
- ii) Let g_1 and g_2 be retractions on F such that $\text{dom}(g_1) \cap \text{dom}(g_2) = \emptyset$ and $(\text{dom}(g_1) \cup \text{dom}(g_2)) \cap (\text{codom}(g_1) \cup \text{codom}(g_2)) = \emptyset$. Then $g_2 \circ g_1$ is a retraction on F . \clubsuit

Evidently there is specific interest in retracts which are minimal.

Definition 2.4. An atomset F is a *core* if its only retract is itself. An atomset F' that is a core, is a *core of* F if it is a retract of F . \dashv

The following two properties are characteristic of cores [27, 35]⁵:

- i) A core of an atomset F is a minimal subset which is homomorphic to F (hence also logically equivalent to F).
- ii) All cores of an atomset F are isomorphic.

If we look again at the atomset $\{p(a, x), p(a, y)\}$, we see that it has two cores: either we retract y on x , resulting in the core $\{p(a, x)\}$, or inversely with the retraction $\{x \mapsto y\}$ we arrive at $\{p(a, y)\}$.

It has been customary to use *core* as an operator to a factbase F , in order to obtain what one would call “its” core. While from the point of view of first order logic and of graph theory, there is sense in considering the core of a factbase as one unique sub-factbase, in this work we want to be able to differentiate between different isomorphic factbases. Especially since we consider a factbase as an atomset, it is indeed common that there is more than one core. Hence in our setting, it can be a bit misleading to use *core* as an operator. Nevertheless when the choice of isomorphic subset of F does not matter, we will use *core* as a (non-deterministic) operator. Therefore we will be using $core(F)$ to choose any subset of F that is a core. On the other hand, when we want to assert that a certain predefined subset F' of F is indeed a core of F , we will utilize bold font, writing $F' \in \mathbf{cores}(F)$, i.e. we denote the set of all (isomorphic) cores of F with $\mathbf{cores}(F)$.

Finally, we introduce a notion which is crucial when we want to manipulate atomsets.

Definition 2.5 (Piece[1]). Let F be an atomset. Every atom $A \in F$ is *connected* to itself. Two atoms $A, A' \in F$ are *connected* (by variables) if there is a sequence $A_1, \dots, A_n \in F$ such that $A_1 = A$, $A_n = A'$ and for every $i < n$ there exists a $z \in \mathbf{var}(A_i) \cap \mathbf{var}(A_{i+1})$. A *piece* in F is a maximal (non empty) set of connected atoms. \dashv

⁵Cores are called *irredundant graphs* in this book.

Note that for every atomset F , the set of pieces in F is a partition of F . Pieces are important because they represent the extent of interdependence among variables in an atomset. This becomes clear when we are investigating the existence of (non-trivial) retracts of a factbase. If a factbase has two or more different pieces and we are looking for possible retractions, we can try firstly with only one of the pieces.

Proposition 2.1 (Retraction by Piece[33]⁶). Let F be an atomset, σ a retraction on F and F' a piece in F . Let $F'' = F \setminus F'$. Then the restrictions $\sigma_{F'}$ and $\sigma_{F''}$ of σ on F' and F'' respectively are retractions on F . Moreover it holds that $\sigma = \sigma_{F'} \circ \sigma_{F''}$. ♣

Example 2: Let $F = \{p(a, x), p(y, b), p(a, b)\}$. Then each singleton subset of F is a piece in F . So $F' = \{p(a, x)\}$ and $F'' = \{p(y, b)\}$ are pieces in F . Let $\sigma = \{x \mapsto b, y \mapsto a\}$. We have that σ is a retraction on F , and in fact $\sigma(F) = \{p(a, b)\}$ is the only core of F . Moreover we can see that $\sigma_{F'} = \{x \mapsto b\}$ and $\sigma_{F''} = \{y \mapsto a\}$ are also retractions on F (but they do not produce cores). ■

2.2 Query Answering over Knowledge Bases

In this section we formalize the (Boolean) query answering problem with existential rules and present the basic (abstract) methods that lead to solutions.

2.2.1 Knowledge Base & Universal Model

So far in this chapter we have mainly discussed atomsets, which are used to represent factual knowledge (corresponding roughly to the concept of a *database*). Ontological knowledge is represented with existential rules. A pair (\mathcal{R}, F) of a ruleset and a (finite) factbase is called a *knowledge base*. We are interested in whether a knowledge base entails a given atomset which is commonly called *Boolean conjunctive query*. In this thesis, with *query* we will mean Boolean conjunctive query.

⁶Pieces correspond to connected components in the *Gaifman graph of the nulls* [33].

The *Boolean conjunctive query entailment problem* is a fundamental problem of interest in the field of existential rules. It can be formulated as follows:

Given a knowledge base (\mathcal{R}, F) and a boolean conjunctive query Q , is it true that $\mathcal{R}, F \models Q$?

In other words, is every model of $\mathcal{R} \cup \{F\}$ (seen as a logical theory) also a model of Q ? While the boolean conjunctive query entailment problem is undecidable for general existential rules [11], we can achieve decidability by restricting the rule language.

A relevant notion is that of a *universal model* [36, 18]. Practically we can conceive first-order structures as possibly infinite factbases [8]. In this regard, a (possibly infinite) atomset S is a *model* of a knowledge base (F, \mathcal{R}) if there is a homomorphism from F to S and for every rule $R = (B, H) \in \mathcal{R}$, if there is a homomorphism π from B to S can be extended to a homomorphism $\pi' \supseteq \pi$ from H to S .⁷ A model U of (F, \mathcal{R}) is *universal* if for every other model S of (F, \mathcal{R}) , there is a homomorphism from U to S . So we can regard a universal model as a “most general” way to expand the initial factbase F so that it satisfies all the rules of \mathcal{R} . If there is a finite universal model of a (F, \mathcal{R}) , then all the universal models are isomorphic.

There are two general approaches to solving the Boolean conjunctive query entailment problem. Either we use the rules to enrich the factbase with all implicit facts that follow from the knowledge base (with the aim of generating a universal model) and then we evaluate the query on this enriched factbase, or we use the knowledge base to rewrite the query in order to arrive at a more easily (or directly) solvable query. Both approaches can be seen as ways of reducing the BCQ entailment problem to a simpler problem (like that of answering a query on a factbase instead of a knowledge base).

⁷therefore it holds that $\pi'_B = \pi$ where π'_B is the restriction of π on B (as defined in Subsection 2.1.2).

2.2.2 Forward Chaining

The process of applying rules on a factbase in order to infer more knowledge is called *forward chaining*. Forward chaining in existential rules is usually achieved via a family of algorithms called *the chase* [37]. The motivation behind such a process is that the result of any chase algorithm is a universal model of the knowledge base and hence the boolean conjunctive query entailment problem is equivalent with asking whether there exists a finite subset of the result of the chase which only by itself logically entails the query [1, 2]. As a consequence, chase termination ensures decidability of the boolean conjunctive query entailment problem. Forward chaining is primarily based on the notion of rule application.

Definition 2.6 (Rule Application). Let F be a factbase, $R = (B, H)$ an existential rule and \bar{z} a set of fresh variables, i.e. a set of variables that is disjoint with the set $\text{var}(F) \cup \text{var}(R)$. Then, R is *applicable to F* if there exists a homomorphism π from B to F . Let σ be a variable renaming such that $\sigma(\text{exv}(R)) \subseteq \bar{z}$ and $\sigma(x) = x$ for $x \notin \text{exv}(R)$. Then the factbase $\alpha(F, R, \pi) = F \cup \sigma(\pi(H))$, is called an *immediate derivation* from F through (R, π) . \dashv

Notice that there can be many different immediate derivations from F with a rule R , since there can be different homomorphisms mapping B to F . This is why it is significant to note that the application of the rule substantiates through the pair (R, π) . These pairs are called *triggers* and hold an important position in this research. Recurring immediate derivation steps, as illustrated in the following example, constitute what we call forward chaining.

Example “GENERIC” 3: Let $F = \{p(a, b), p(c, d), r(e)\}$ and \mathcal{R} the following set of rules:

$$\begin{aligned} R_1 &= p(x, y) \wedge r(z) \rightarrow p(y, z) \\ R_2 &= p(x, y) \wedge p(y, z) \rightarrow \exists u p(z, u) \\ R_3 &= p(x, y) \wedge p(x, z) \rightarrow p(y, z) \end{aligned}$$

Then $\pi_1 = \{x \mapsto a, y \mapsto b, z \mapsto e\}$ is a homomorphism that maps the body of R_1 to F . Hence an immediate derivation from F through

(R_1, π_1) is $F_1 = F \cup \{p(b, e)\}$. Then we can apply the rule R_2 with $\pi' = \{x \mapsto a, y \mapsto b, z \mapsto e\}$, obtaining $F_2 = F_1 \cup \{p(e, u_0)\}$ (where u_0 is a fresh variable). Afterwards, we can take $\pi'' = \{x \mapsto b, y \mapsto e, z \mapsto u_0\}$ and reapply R_2 to get $F_3 = F_2 \cup \{p(u_0, u_1)\}$. Subsequently we can either continue in a similar fashion, by reapplying R_2 to the last added atoms, or chose any suitable pair of atoms of F_3 to apply R_1 or R_3 . Evidently, those choices lead to factbases with different features which can include semantical disagreement if the forward chaining is not exhaustive.⁸ ■

The apparent non-determinism of this definition of rule application can be reduced to a managable scale, if we impose for example a breadth-first prioritization in the order with which the rules are applied. As presented for instance in [1, 8, 38], we can extend the operator α to be applicable to a knowledge base, resulting in the expansion of the initial factbase with all the facts which can be directly inferred after only one rule application from the ruleset. In particular given a knowledge base $K = (F, \mathcal{R})$, we define the set of (*immediate*) *triggers* on K as the following set of pairs of rules and homomorphisms:

$$\mathsf{T}(K) := \{((B, H), \pi) \mid (B, H) \in \mathcal{R}, \pi(B) \subseteq F\}$$

and using this notion we can define the one-step saturation of F by \mathcal{R} as

$$\alpha(F, \mathcal{R}) := F \cup \bigcup_{((B, H), \pi) \in \mathsf{T}(K)} \sigma^{safe}(\pi(H))$$

where σ^{safe} is a renaming such that all existential variables of H are mapped to *fresh variables* (i.e. variables that do not appear in K and are exclusively used in one particular rule application). We can then reapply the α -operator to the obtained result a number of times (possibly infinite), denoting the number of applications with an index:

⁸We will formally define exhaustivity in Chapter 4. It refers to applying exhaustively and recursively all possible rules.

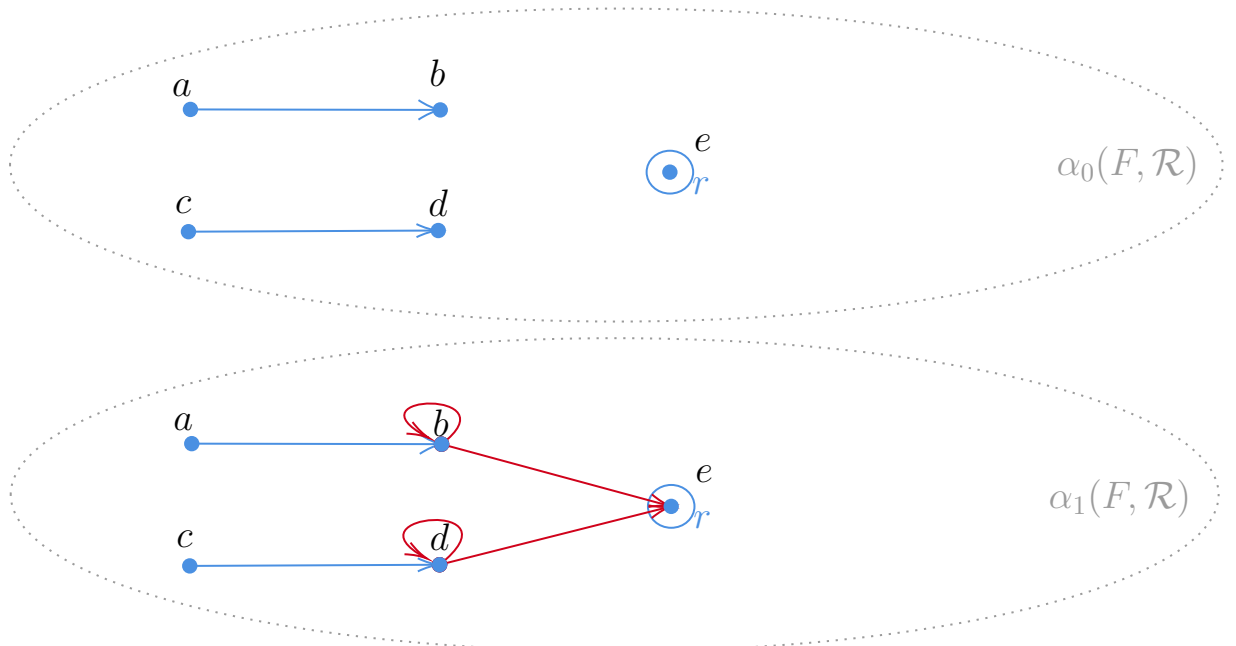
$$\begin{aligned}
\alpha_0(F, \mathcal{R}) &:= F, \\
\alpha_1(F, \mathcal{R}) &:= \alpha(F, \mathcal{R}), \\
\alpha_2(F, \mathcal{R}) &:= \alpha(\alpha(F, \mathcal{R}), \mathcal{R}), \\
&\vdots \\
\alpha_n(F, \mathcal{R}) &:= \alpha(\alpha_{n-1}(F, \mathcal{R}), \mathcal{R}) \text{ and} \\
\alpha_\infty(F, \mathcal{R}) &:= \bigcup_{n \in \mathbb{N}} \alpha_n(F, \mathcal{R})
\end{aligned}$$

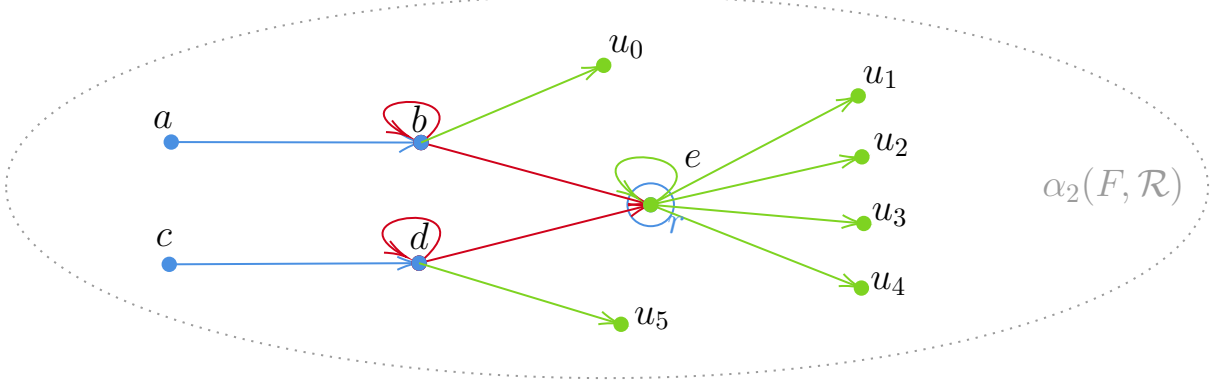
We call $\alpha_i(F, \mathcal{R})$ the *result* of forward chaining of *depth* i on (F, \mathcal{R}) . And this procedure can be optimized for algorithmic use by adding the condition that each pair of rule and homomorphism (i.e. trigger) is used at most once in the entire (breadth-first) forward chaining process. In Chapter 4 we will define this mechanism on top of the notion of *derivation*, which has the additional property that a particular order of rule applications is specified.

Example “GENERIC” 4 (continued from Example 3): We use again the knowledge base (F, \mathcal{R}) where $F = \{p(a, b), p(c, d), r(e)\}$ and \mathcal{R} comprises three rules:

$$\begin{aligned}
R_1 &= p(x, y) \wedge r(z) \rightarrow p(y, z) \\
R_2 &= p(x, y) \wedge p(y, z) \rightarrow \exists u p(z, u) \\
R_3 &= p(x, y) \wedge p(x, z) \rightarrow p(y, z)
\end{aligned}$$

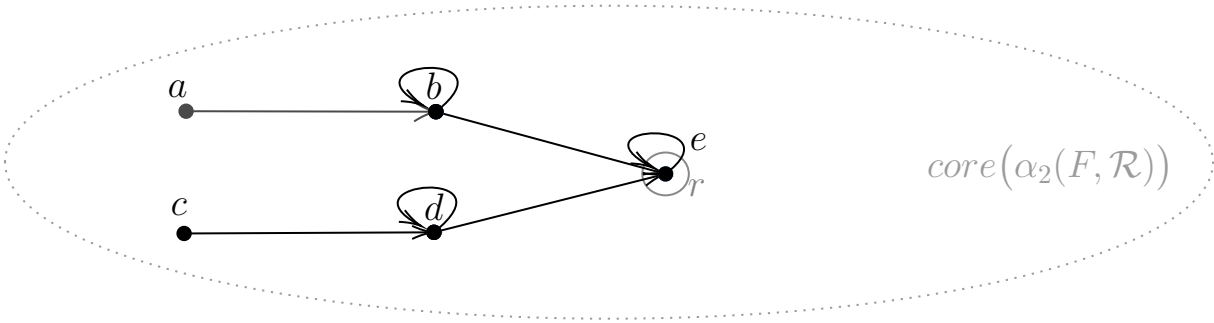
To provide an intuitive understanding, below we illustrate the labelled graphs corresponding to $\alpha_0(F)$, $\alpha_1(F)$ and $\alpha_2(F)$.





We have removed the label p from the edges, since there is no other binary predicate, for the sake of readability. Blue atoms are part of F , red atoms are added in $\alpha_1(F, \mathcal{R})$ (with two applications of R_1 and two applications of R_3) and green atoms are added in $\alpha_2(F, \mathcal{R})$ (by applying R_3 and R_2). Notice how in the second step there is an explosion of new variables, since the existential rule R_2 of \mathcal{R} is applicable to $\alpha_1(F, \mathcal{R})$ through many different homomorphisms, i.e. R_2 is included in many of the (immediate) triggers on $(\alpha_1(F, \mathcal{R}), \mathcal{R})$.

Finally we can see that the following atomset is a core of $\alpha_2(F, \mathcal{R})$:



Moreover, by calculating $\alpha_3(F, \mathcal{R})$ ⁹ we can see that it has the same core as $\alpha_2(F, \mathcal{R})$, which implies they are logically equivalent, hence all rule applications on $\alpha_2(F, \mathcal{R})$ produce redundant information. ■

We conclude this section with the definition of an abstract class of existential rules, which provides a condition for the termination of forward chaining.

⁹which amounts to

- i. duplicating all atoms that include a variable with twin fresh variables, i.e. for every atom that includes a variable u_x , add a variation of this atom with variable u'_x , and
- ii. adding for every variable u_x of $\alpha_2(F, \mathcal{R})$, the three atoms $r(u_x)$, $p(u_x, e)$ and $p(u_x, u_y)$, where u_y is a new variable.

Within this abstract class, the boolean conjunctive query entailment problem is decidable, however membership of a ruleset in this class is an undecidable problem [1].

Definition 2.7. A ruleset \mathcal{R} is a *finite expansion set (fes)* if for every factbase F there is a $k \in \mathbb{N}$ such that $\alpha_k(F, \mathcal{R}) \equiv \alpha_{k+1}(F, \mathcal{R})$. \dashv

Notice that since for all $k \in \mathbb{N}$, $\alpha_k(F, \mathcal{R})$ is a finite factbase, we know that a knowledge base (F, \mathcal{R}) where \mathcal{R} is fes always has a finite universal model.

To circumvent the undecidability of the problem of membership in the above abstract class, several recognizable subclasses of fes have been identified, which are almost always based on some notion of acyclicity [39]. This division into those recognizable classes is not pertinent for this dissertation, thus we do not present any of them.

A main interest in this work is the exploration of the circumstances under which a given ruleset enjoys the property of having a general bound on the depth of forward chaining based on it, independently of the initial factbase. We will delve into different related notions of boundedness. The concept of *core-boundedness* or simply *boundedness* lies at the root of this research.

Definition 2.8 (Boundedness for Existential Rules[38]). A ruleset \mathcal{R} is said to be *bounded* if there exists a $k \in \mathbb{N}$ such that for every factbase F , $\alpha_k(F, \mathcal{R}) \models \alpha_{k+1}(F, \mathcal{R})$. \dashv

As simple “primitive” examples, let us take the singleton rulesets $\mathcal{R}_1 = \{p(x, y) \rightarrow q(x, y)\}$ and $\mathcal{R}_2 = \{p(x, y) \rightarrow \exists z p(y, z)\}$. We can see that \mathcal{R}_1 is bounded because its only rule does not generate any atoms on which a new rule application can be based. On the other hand \mathcal{R}_2 is not bounded since if we start from an atom isomorphic with the body of its rule, we can produce an infinite directed chain of new variables connected with p .

This concept of boundedness, although originally introduced in the context of the universal relation database model [40], was initially mainly studied in fragments and variations of datalog [29, 30, 31, 25]. There it was mainly linked to the potentiality of eliminating recursion: detecting bounded datalog rulesets

is a robust optimization technique since such a ruleset can be transformed to a ruleset that completely saturates any factbase with all possible new facts in a single breadth-first rule-application step. In the next chapter we present some of the results of this research.

2.2.3 Backward Chaining

The utilization of the knowledge base on the query in order to produce a new query is called *backward chaining*. This paradigm can be employed in many different situations. In our case, we only use the rules to rewrite the query into a disjunction of conjunctive queries and then verify whether the factbase alone entails this disjunction (hence at least one of those queries) [4]. This provides a solution to the boolean conjunctive query entailment problem, when the rewriting process terminates. We can represent a finite disjunction of conjunctive queries as a union of atomsets (queries), this is why it is known as a *union of conjunctive queries (UCQ)*. So starting from a factbase F , a ruleset \mathcal{R} and a query Q , we are searching for a UCQ \mathcal{Q} such that $F \models Q$ (where we view the UCQ as a disjunctive first order logic formula) if and only if $F \cup \mathcal{R} \models Q$. This is particularly advantageous if we have a big volume of data (big factbase) as this procedure is independent of the factbase.

Before discussing the rewriting process, we introduce the relevant abstract class of rules. Obtaining the desired finite disjunction of boolean conjunctive queries is not possible with any ruleset. A ruleset which always produces finite UCQs as rewritings of queries is called *finite unification set* [1, 41]:

Definition 2.9. A ruleset \mathcal{R} is a *finite unification set (fus)* if for every query Q and factbase F , there exists a (finite) UCQ \mathcal{Q} such that $F \cup \mathcal{R} \models Q$ if and only if $F \models \mathcal{Q}$. \dashv

The property that a ruleset is fus is also called *UCQ-rewritability* in the literature and it is also known to be equivalent with *first-order rewritability* [42, 43, 44]. A single rewriting of a query intuitively corresponds to a factbase which can produce a specialization of the query by a single application of one rule. In particular, when a single rewriting step produces a query Q'

from Q by a rule R , it holds that a factbase F entails Q' if and only if there is an application of R to F that produces a factbase F' that entails Q . A series of rewritings corresponds to more complex ways (involving more rule applications) to produce a specialization of the query from some initial factbase.

Example 5: Intuitive Rewriting. We assume our ruleset includes only the rule $R = p(x, y) \wedge p(y, z) \rightarrow p(x, z)$. This rule is the typical example of a *transitive* rule. Such rules lead to infinite rewritings. Indeed, if we take $Q = \{p(a, b)\}$ as a query and we ask whether there is some factbase which could produce Q with the application of R , we find that one application of R on the factbase $Q' = \{p(a, z_0), p(z_0, b)\}$ can produce Q . This establishes that $Q' = \{p(a, z_0), p(z_0, b)\}$ is a rewriting of Q . However couldn't Q' itself be produced by some other factbase by applying R ? Indeed, we find that $Q'' = \{p(a, w_0), p(w_0, z_0), p(z_0, b)\}$ is a rewriting of Q' hence also a rewriting of Q . Furthermore, Q' does not entail Q'' . This process can go on forever. Lastly, we stress that the query Q can be considered a rewriting of itself. This is because in the end what we want is to produce all the queries that can be entailed by a factbase F whenever the knowledge base $(\{R\}, F)$ entails Q . Hence, the first such query is Q itself.¹⁰ ■

In the above example we use a rule which does not have any existential variable in the head. In such a case it is easy to see that rewriting can be formally defined using most general unifiers (introduced in subsection 2.1.2) of the query with the head of the rule. However in the general case of existential rules there are some additional constraints that need to be satisfied in order to achieve a consistent rewriting. In the Appendix (Section (A)) we provide details concerning how query rewriting works in an existential rule setting.

¹⁰The fact that we used the transitivity rule in an example portraying the transitivity of the rewriting relation (every rewriting of a rewriting is a rewriting) is rather coincidental.

3 Boundedness in the Literature

In this chapter we outline the main results of current and past research on the characterization of boundedness for different fragments of existential rules, in particular the Datalog fragment .

3.1 Datalog

The datalog language was the first fragment of existential rules to be extensively studied in the context of query answering. It was motivated as a query language for databases which adds recursivity to the power of relational algebra [6]. We remind that an existential rule is *datalog* if there is no existentially quantified variable in the head and a ruleset with datalog rules is a *datalog* rule-set. Moreover, we assume every rule to have a single-atom head. This is not a real constraint, as every set of datalog rules that includes multiple atoms in the head can be trivially transposed to an equivalent set of rules with just one atom in the head. That is done by having multiple versions of the rules, one for each different atom in the head of the original rule while all versions keep the original body as is. For example $p(x) \rightarrow q(x) \wedge r(x)$ is transformed to $p(x) \rightarrow q(x)$ and $p(x) \rightarrow r(x)$.

One major feature that distinguishes datalog from the rest of the (sub-)languages of existential rules, is that in forward chaining, the factbase is not enriched with new individuals. Besides, in the original datalog framework, facts are always assumed to be grounded, i.e. there are no variables in the factbase. We will

use the term *database* to refer to a factbase that contains no variables. Hence when we say *datalog knowledge base*, it will be implied that the factbase is indeed a database. Given a datalog ruleset, the predicates found in the head of some rule are called *intentional database (IDB) predicates* and the rest of the predicates are called *extensional database (EDB) predicates*. Accordingly the atoms of a rule with an IDB predicate are called its *IDB atoms*, while all the rest of the atomic formulas appearing in the ruleset are its *EDB atoms*. This distinction of predicates into two categories is important because it gave birth to two different concepts of boundedness. In the context of datalog research, what we call *boundedness* is referred to as *uniform boundedness*. Thus we will say that datalog ruleset \mathcal{R} is said to be *uniformly bounded* if there exists a $k \in \mathbb{N}$ such that for every factbase F , $\alpha_k(F, \mathcal{R}) = \alpha_{k+1}(F, \mathcal{R})$. But there is one more type of boundedness, based on the following notion:

Definition 3.1. Let \mathcal{R} be a datalog ruleset. A database F is *extensional* for \mathcal{R} if only EDB predicates of \mathcal{R} appear in it. \dashv

We see that if a ruleset \mathcal{R} is given, only a subset of all the possible databases F are extensional for \mathcal{R} .

Definition 3.2 ([31, 25]). A datalog ruleset \mathcal{R} is *program bounded* if there is a $k \in \mathbb{N}$ such that for every database F that is extensional for \mathcal{R} , holds that $\alpha_{k+1}(F, \mathcal{R}) = \alpha_k(F, \mathcal{R})$. \dashv

By definition it is evident that uniform boundedness of a ruleset implies its program boundedness. The inverse is not true as can be shown in the following example:

Example 6: We take the ruleset $\mathcal{R} = \{R_1, R_2\}$ that contains the rules, $R_1 = p(x, y) \wedge p(y, z) \rightarrow p(x, z)$ and $R_2 = q(x) \wedge q(y) \rightarrow p(x, y)$. We find that \mathcal{R} is program bounded since p is IDB and so we can only have q -atoms in the database. So for any extensional database F , at first R_2 will be applied on all possible q -atom pairs, creating all possible p -atoms, so then any application of R_1 will not produce any new atom. However, since R_1 is a transitive rule, by choosing a chain of 2^n atoms with predicate p as database, the breath-first

forward chaining stops at depth n . So the breadth-first forward chaining does not have a bound in its depth with an arbitrary database. Therefore the set is not uniformly bounded. ■

Another relevant notion is that of *predicate boundedness*. In most publications it is introduced as a sub-problem of program boundedness, however here we follow the more general, independent definition of predicate boundedness as in [26], i.e. we present two definitions for *program predicate boundedness* and *uniform predicate boundedness*. For the next definition we use the following notation: if F is a factbase and p a predicate, we denote with $F|_p$ the set of all the p -atoms of F .

Definition 3.3. Let \mathcal{R} be a datalog ruleset. A predicate p is *uniformly bounded* in \mathcal{R} if there exists a $k \in \mathbb{N}$ such that for every database F , $\alpha_{k+1}(F, \mathcal{R})|_p = \alpha_k(F, \mathcal{R})|_p$. Similarly p is *program bounded* if there exists a $k \in \mathbb{N}$ such that for every database F that is extensional for \mathcal{R} , $\alpha_{k+1}(F, \mathcal{R})|_p = \alpha_k(F, \mathcal{R})|_p$. ─

The subsequent statement follows from the above definition:

Proposition 3.1. A datalog ruleset \mathcal{R} is (uniformly) bounded if and only if every (IDB) predicate p that appears in \mathcal{R} is uniformly bounded. Furthermore \mathcal{R} is program bounded if and only if every (IDB) predicate p in \mathcal{R} is program bounded. ♣

At this point we make a summary of the known decidable and undecidable subclasses of the boundedness problems. We first note that as a consequence of proposition 3.1, the problems of deciding program and uniform boundedness can be reduced to those of program predicate boundedness and uniform predicate boundedness respectively. Furthermore, it can be shown that:

Proposition 3.2 ([45]). The problem of determining whether a datalog ruleset is uniform bounded can be reduced to the problem of determining whether a datalog ruleset is program bounded.

Proof: Suppose that we have an algorithm to check program boundedness. Let \mathcal{R} be a datalog ruleset. For every IDB predicate p in \mathcal{R} we create a rule

$R_p = p'(\bar{x}) \rightarrow p(\bar{x})$ where p' is a fresh predicate that does not appear in \mathcal{R} . Then the ruleset $\mathcal{R}' := \mathcal{R} \cup \{R_p \mid p \text{ is an IDB predicate of } \mathcal{R}\}$ is program bounded if and only if \mathcal{R} is uniformly bounded. This can be verified by remarking that every chase of \mathcal{R} on F is equivalent with the chase of \mathcal{R}' on $\alpha_1(F', \mathcal{R})$, where F' occurs by replacing each predicate p in F with its respective p' . \square

Unfortunately boundedness of a datalog ruleset is undecidable in general as well as for prominent subclasses. But before we refer to the undecidability results in detail we discuss positive results (i.e. decidability) which have been shown only for particular restricted languages. We provide the following definitions to clarify the decidability results: a rule is called *recursive* if its body includes at least one IDB predicate. Note that a datalog ruleset might include recursive rules while being non-recursive. A datalog rule is *datalog linear*, or in short *linear_d*, if its body contains at most one IDB predicate. Notice that what is called *linear_d* in datalog does not correspond to the definition of linear existential rules, hence we use the subscript “d” to distinguish the terminology (in particular a *linear_d* rule is not necessarily a linear existential rule¹). A datalog ruleset is *linear_d* if it only includes *linear_d* rules. A datalog ruleset is *monadic* if every IDB predicate is unary. But we call a datalog ruleset *binary*, *ternary* or *of arity n* if the maximum arity of its predicates is 2, 3 or n respectively.

Decidability of program boundedness (and hence uniform boundedness as well) has been proved for monadic rulesets [31, 46] and for rulesets with a single *linear_d* recursive rule if the IDB is binary [47]. Furthermore, program boundedness (and hence uniform boundedness as well) is also decidable for binary rulesets where each IDB appears only once in the head of a rule [26]. Lastly, program boundedness for *chain rulesets*² has been shown to be decidable [48].

However undecidability is shown for uniform boundedness of ternary *linear_d* rulesets and single recursive rule ternary rulesets (which subsume ternary rulesets in general) [45] and also for single rule rulesets [45]. Pro-

¹e.g. $p(x, y) \wedge q(x) \rightarrow p(y, x)$ is a *linear_d* rule but not a linear existential rule.

²A *chain rule* is of the form $p_1(\bar{x}_0, \bar{x}_1) \wedge p_2(\bar{x}_1, \bar{x}_2) \wedge \dots \wedge p_n(\bar{x}_{n-1}, \bar{x}_n) \rightarrow q(\bar{x}_0, \bar{x}_n)$, where $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ are distinct variables, or distinct tuples of variables of the same arity.

gram boundedness is shown to be undecidable for linear_d binary rulesets [30]. Finally, uniform and program boundedness are undecidable for rulesets consisting of one linear_d recursive rule and one non-recursive rule [45]. It is not yet proven whether the uniform boundedness of binary rulesets, linear_d binary rulesets and rulesets consisting of a single linear_d rule is decidable or not.

3.2 Existential Rules

The research around boundedness for Existential Rules is a new field of study, hence there exist only a few publications, approaching the problem from different angles. At first we must mention that there is a number of studies that incorporate some form of the term “bounded” in their terminology, whilst being (semantically) unrelated to our concept of boundedness. In particular, some publications designate a ruleset “bounded” if the size of skolem terms that can be created in the factbase is bounded by an integer function whose input is the size of the ruleset [49, 50]. Therefore they do not include any results directly related to boundedness as we define it.

A question that arises immediately after discussing the division between program boundedness and uniform boundedness in datalog, is whether this could be extended to general existential rules. Interestingly, the *data exchange setting*, introduced in [36] makes the same distinction between the EDB and IDB predicates (albeit calling them *source & target schema* respectively). There is a decent volume of research related to the data exchange setting, including some relevant to our work like [33], which concerns the calculation of the core of an atomset. However there has been no research on the concept of boundedness in this setting, showing an interesting prospective for further study.

Although our investigation started with the adoption of Definition 2.8 for boundedness, this is only one of the many ways to extend what is defined as uniform boundedness in datalog to a corresponding notion for existential rules. That is because the presence of variables motivates a number of different methods to manipulate the forward chaining process, all of which are related in one way or another to reducing redundancy. As mentioned in the previous chapters,

one of the prevalent perspectives of early research on datalog boundedness is its connection with recursivity in the ruleset [6, 25, 26]. In existential rules, some recursive rulesets can be bounded because the rules that can be applied recursively are redundant with respect to other rules. In this vein, there is an article [51], which provides preliminary definitions for several forms of boundedness for existential rulesets, using the notion of redundancy in the ruleset³. The framing of boundedness in relation with redundancy in the ruleset seems as another approach with great potential for future research, however in this dissertation we follow a different path.

The perspective on which we based our analysis, assumes a random (black-box) existential ruleset. Our interest concerns the imposition of conditions on the forward chaining with the goal of reducing redundancy in the resulting factbase. To this end, there is a number of different algorithms that have been introduced, collectively called *the chase*, which employ different techniques to filter out some of the possible redundant atoms that can be introduced in the forward chaining process. In the following chapter we will define the *oblivious chase*, the *semi-oblivious chase*, *restricted chase* and the *core chase* (among other variants). Each one of those algorithms relates to a different kind of boundedness⁴, but all the different definitions collapse to the same notion when we work in a datalog setting. Hence, the general undecidability result from datalog propagates to existential rules.

There is one specific class of existential rules, which is by definition bounded, namely the “acyclic graph of rule dependency” class, in short *aGRD* [1]. This class is defined by the acyclicity of a graph (called the graph of rule dependency) whose nodes are the rules and edges translate the fact that the application of the first rule may trigger a new application of the second rule. A ruleset that belongs to the *aGRD* class is bounded by the length of a maximal path in its graph of rule dependencies. All other known concrete *fes* classes generalize datalog (see [39]). The landscape is less clear concerning concrete *fus* classes. One specific *fus* class for which there are positive results is that of

³What is called *strong boundedness* in this article corresponds to what we call *fus*.

⁴which always concerns a bound to the depth of the forward chaining achievable using a certain ruleset and any initial factbase.

linear existential rules, which are rules that have only one atom in the body.

In a brief break from our black-box approach, we will arrive to positive results concerning boundedness in the special case of linear existential rules. We will show that with linear rules and for certain chase variants, chase termination⁵ and boundedness are equivalent notions. This allows us in a direct manner to extrapolate results concerning boundedness from results concerning chase termination. And thankfully, chase termination for linear rules has been researched a lot. We know that oblivious & semi-oblivious [20] and core chase [21] termination are decidable for linear existential rules. In the case of *extra linear rules*, where both body and head comprise of only one atom each, restricted chase termination is also shown to be decidable [22].

Most of our research focuses on a specialization of the problem of boundedness, called k -boundedness, where the depth k is predetermined. To the best of our knowledge this is the first research of k -boundedness (except our own publication [52]⁶, whose results are incorporated in this thesis).

⁵The *chase termination* problem for a ruleset \mathcal{R} , asks whether it is true that for every factbase F , the chase algorithm starting from the knowledge base (F, \mathcal{R}) terminates. It is also called *all-instance* termination.

⁶There exists an updated revised version of this article [53], but of course the most complete account of this work is within this thesis.

4 The Chase

In Section 2.2 we introduced the concept of forward chaining. In this chapter we focus on the particularities of algorithms designed to carry out this task, which are collectively called *chase* algorithms. The chase is a fundamental tool for reasoning on rule-based knowledge bases and a considerable literature has been devoted to its analysis, approaching it from a variety of presupposed terminological and notational background. A central part of this work is the establishment of a general framework whereby several known chase algorithms can be represented. In addition, this framework provides a basis for the possible definition of new chase algorithms. And indeed, as an optimization that emerges from our research around boundedness, we do define new chase algorithms (in this as well as in the following chapter). The main formal framework is set in Section 4.1, whereas in Sections 4.2 and 4.3 we present several chase variants. In Section 4.4 we discuss criteria of comparison as well as the breadth-first approach in relation with the specified chase variants and finally in Section 4.5 we expand the theoretical tools at our disposal for the analysis of the chase and we show several fundamental properties of chase algorithms, which will prove to be valuable in Chapter 5.

4.1 Derivations

In Definition 2.6 we introduce the formal term “immediate derivation”. Hereafter we will get into more detail and we will employ this concept to designate our framework for the study of the chase. At first we revisit this definition, adding a little bit more terminology, as well as a very useful syntactic specifi-

cation.

Let F be a factbase and $R = (B, H)$ be an existential rule. Recall that if there exists a homomorphism π from the body B of R to F , we say that R is *applicable* on F via π . Then, the pair (R, π) is called a *trigger*, usually denoted with $\tau = (R, \pi)$, and we will also say that τ is *applicable* on F . Moreover, to simplify notation in cases where R, π, B and H are not specified, $\pi(B)$ is called the *support* of τ and is denoted with $\text{sp}(\tau)$. And so by definition it holds that τ is applicable on F if and only if $\text{sp}(\tau) \subseteq F$.

Triggers will play a crucial role in the following of this work. Notably, each time that we apply a rule, we are applying a trigger. Furthermore, during the execution of any chase algorithm, each trigger is applied at most once. This uniqueness of each trigger in a chase algorithm, can be used as a key to trace the new variables added to the factbase¹. That is achieved as follows: the application of the rule $R = (B, H)$ to F with trigger $\tau = (R, \pi)$ results to the factbase $F \cup \pi^s(H)$ which (in accordance with the definition in Chapter 2) is called an *immediate derivation* from F through τ . With π^s we denote an extension of π which maps all existentially quantified variables in H to fresh variables which are indexed by the trigger τ . In particular, for each existential variable z in H we have that $\pi^s(z) = z_\tau$. This fixed way to choose a new fresh variable allows us to always produce the same atoms when applying the same trigger on a different context and will be very useful when comparing forward chaining with the same ruleset but different factbase. Although this method is not genuinely novel since it follows the skolemization paradigm, to the best of our knowledge the only work which hints towards a similar approach was published the previous year [54]. Again, to simplify notation in cases where R, π, B and H are not specified, $\pi^s(H)$ is called the *output* of τ and is denoted with $\text{op}(\tau)$. As a result we can represent an immediate derivation from F through τ as $F \cup \text{op}(\tau)$. Moreover, the (fresh) variables introduced (and indexed by) a trigger τ are called *nulls* or *new* variables. Accordingly, the set of new variables introduced by τ is denoted with $\text{nul}(\text{op}(\tau))$. Lastly, a trigger

¹The newly introduced variables are also called “nulls” in the relative literature and in some cases they are regarded as new constants.

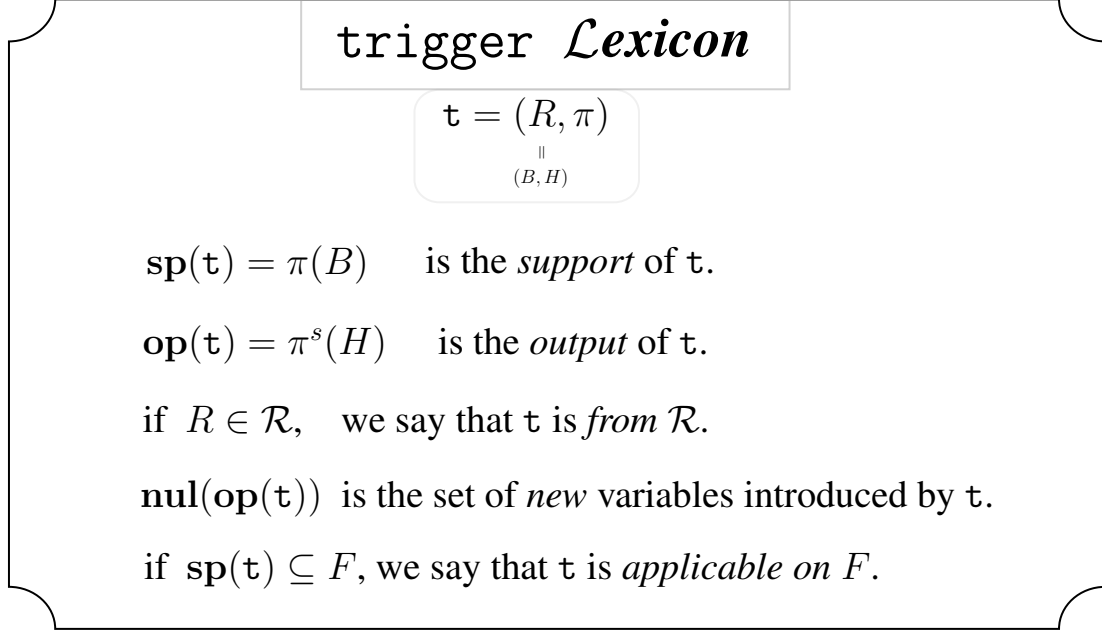


Figure 4.1: The Trigger Lexicon.

(R, π) where $R \in \mathcal{R}$ is said to be a trigger *from* \mathcal{R} . In Figure 4.1 we have summed up the main notions related to triggers.

The chase is built upon the notion of derivation, which consists of the repeating application of rules from a certain ruleset to a factbase which is evolving with every rule application. We provide a novel definition of derivation which is meant to be general enough such that all the chase algorithms that are known to us and produce universal models of the knowledge base (namely *oblivious*, *semi-oblivious*, *restricted*, *core*, *parallel*, *frugal* and *equivalent chase*) can be expressed as specifications of this definition, i.e. as derivations that satisfy certain additional constraints. Some of those algorithms remove redundant atoms from the factbase. Others just avoid the application of certain redundant triggers. Some of the algorithms are sequentially applying the rules, others carry out parallel rule applications. Therefore our definition of derivation is as general as possible and it outlines those central operations of forward chaining, which is the application of triggers and the expulsion of redundant information.

Here is how a derivation is formed: we start from a knowledge base (F, \mathcal{R}) . The primary task is to apply rules from \mathcal{R} on an evolving factbase Z which at first is instantiated as F . Each rule application corresponds to a unique trigger, so we are not allowed to repeat triggers within the same derivation. After ap-

plying a trigger \mathfrak{t} from \mathcal{R} on Z , a new factbase is produced $F' = Z \cup \text{op}(\mathfrak{t})$, i.e. F' is the immediate derivation from Z through \mathfrak{t} . A secondary (optional) task is to eliminate redundance from our factbase. Therefore we search for a retract Z' of F' in order to remove redundant variables (and their respective atoms). Once we obtain Z' we search for a new trigger \mathfrak{t}' that is applicable on Z' and the procedure goes on like this. Notice that Z' is not necessarily a core of F' , but may be any kind of retract.

Those are the main functionalities of a derivation. Yet before presenting a formal definition, we introduce one more feature which we must include in our definition. That is the possibility of parallel rule applications: suppose that the triggers $\mathfrak{t}_1, \dots, \mathfrak{t}_n$ are applicable on Z . Then we can apply all of them at once, producing $F' = Z \cup \text{op}(\mathfrak{t}_1) \cup \dots \cup \text{op}(\mathfrak{t}_n)$. We could of course also apply some of them, or just one of them (as we said, the purpose of our definition is that it is as general as possible). After the parallel application of those triggers, we search again for a retract Z' of F' .

Notice how there are three components in the process described above. A trigger is applied on an *active* factbase, producing a *transitory* factbase, which is then retracted to form a new active factbase. As a result it is suited to represent a derivation as a sequence of triples $(\mathfrak{t}_i, F_i, Z_i)$ of triggers, transitory factbases and active factbases. At every step i of the derivation, the transitory factbase is the immediate derivation from the previous active factbase Z_{i-1} through \mathfrak{t}_i and the active factbase Z_i is either unaffected ($Z_i = Z_{i-1}$), or it is the image of a retraction from the union of all the transitory factbases produced from the previous active factbase Z_{i-1} . This retraction can simply be the identity (hence retaining the whole union as the new active factbase). Here is the formal definition:

Definition 4.1 (Derivation). Given a factbase F and a ruleset \mathcal{R} , a *derivation* \mathcal{D} from (F, \mathcal{R}) , is a (possibly infinite) sequence of triples $D_0 = (\mathfrak{t}_0, F_0, Z_0), D_1 = (\mathfrak{t}_1, F_1, Z_1), D_2 = (\mathfrak{t}_2, F_2, Z_2), \dots$ where $\mathfrak{t}_0 = \emptyset$, $F_0 = Z_0 = F$ and for every $i > 0$ holds that

- i) F_i is an immediate derivation from Z_{i-1} through a new trigger \mathfrak{t}_i (i.e. for

all $i \neq j$ we have $\mathbf{t}_i \neq \mathbf{t}_j$),

- ii) if $Z_i \neq Z_{i-1}$, then Z_i is a retract of $Z_{i-1} \cup \mathbf{op}(t_{\ell+1}) \cup \dots \cup \mathbf{op}(\mathbf{t}_i)$, where ℓ is the minimal number with the property that for every $j \in \{\ell, \dots, i-1\}$ holds that $Z_j = Z_{i-1}$.²

Each D_i is called a *derivation triple* or an *element* of \mathcal{D} . The atomsets F_i and Z_i are called *transitory* and *active* factbases respectively. We denote the set of atoms *produced* in \mathcal{D} with $F^{\mathcal{D}} = \bigcup_i F_i$. If \mathcal{D} is finite, then the final active factbase is denoted with $Z^{\mathcal{D}}$. Given a derivation \mathcal{D} as above, the sequence of triggers *associated* with \mathcal{D} is $\mathbf{trig}(\mathcal{D}) = \mathbf{t}_1, \mathbf{t}_2, \dots$. Finally, an atom A is *produced* by \mathbf{t}_i (in \mathcal{D}) if $A \in F_i \setminus Z_{i-1}$ and i is minimal for that property³. \dashv

The notation $\mathcal{D} = (\mathbf{t}_*, F_*, Z_*)$ will be frequently utilized to specify a derivation. Naturally, if $\mathcal{D} = D_0, D_1, \dots, D_i, D_{i+1}, \dots$ is a derivation, then we call the derivation $\mathcal{D}' = D_0, D_1, \dots, D_i$ a *prefix* of \mathcal{D} , whereas \mathcal{D} is an *extension* of \mathcal{D}' . Additionally, we consider every finite derivation to be a prefix of itself. The variables introduced in a derivation are referred to as *new* variables, and for any subset F' of $F^{\mathcal{D}}$ we can write $\mathbf{nul}(F') := \mathbf{var}(F') \setminus \mathbf{var}(F)$. Depending on the context, $\mathbf{trig}(\mathcal{D})$ might also be considered as a set of triggers (as opposed to a sequence of triggers). Note that by the above specification every atom A that is produced by \mathbf{t} belongs to the output $\mathbf{op}(\mathbf{t})$. However the converse is not true as the output $\mathbf{op}(\mathbf{t})$ might include atoms which are have already been produced by a previous trigger. And according to Definition 4.1, an atom is produced at most once (by at most one trigger) in a derivation.

We will represent derivations with tables where each line corresponds to an element of the derivation. Here is a first example of a derivation:

Example 7: Let $F = \{r(a)\}$ and \mathcal{R} be the following set of rules:

$$R_1 = r(x) \rightarrow \exists y p(x, y)$$

$$R_2 = r(x) \wedge p(x, y) \rightarrow p(y, x)$$

$$R_3 = p(x, y) \wedge p(y, x) \rightarrow r(y) \wedge p(y, y)$$

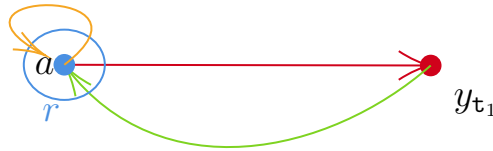
²i.e. $\{Z_\ell, Z_{\ell+1}, \dots, Z_{i-1}\}$ is the maximal set of consecutive equal active factbases before Z_i , so $Z_\ell = Z_{\ell+1} = \dots = Z_{i-1}$. Notice that $Z_{i-1} \cup \mathbf{op}(\mathbf{t}_{\ell+1}) \dots \cup \mathbf{op}(\mathbf{t}_i)$ can also be written as $Z_\ell \cup \mathbf{op}(\mathbf{t}_{\ell+1}) \dots \cup \mathbf{op}(\mathbf{t}_i)$ and also as $F_{\ell+1} \cup F_{\ell+2} \cup \dots \cup F_i$.

³ $i = \min\{j \in \mathbb{N} \mid A \in F_j \setminus Z_{j-1}\}$.

The following table outlines the derivation \mathcal{D} of length 5 from (F, \mathcal{R}) (where the lines represent the derivation triples $D_0, D_1, D_2, D_3, D_4, D_5$):

\emptyset	$F_0 = F$	$Z_0 = F_0$
$\mathfrak{t}_1 = (R_1, \{x \mapsto a\})$	$F_1 = Z_0 \cup \{p(a, y_{\mathfrak{t}_1})\}$	$Z_1 = F_1$
$\mathfrak{t}_2 = (R_2, \{x \mapsto a, y \mapsto y_{\mathfrak{t}_1}\})$	$F_2 = Z_1 \cup \{p(y_{\mathfrak{t}_1}, a)\}$	$Z_2 = F_2$
$\mathfrak{t}_3 = (R_3, \{x \mapsto y_{\mathfrak{t}_1}, y \mapsto a\})$	$F_3 = Z_2 \cup \{p(a, a)\}$	$Z_3 = F_0 \cup \{p(a, a)\}$
$\mathfrak{t}_4 = (R_2, \{x \mapsto a, y \mapsto a\})$	$F_4 = Z_3$	$Z_4 = Z_3$
$\mathfrak{t}_5 = (R_3, \{x \mapsto a, y \mapsto a\})$	$F_5 = Z_3$	$Z_5 = Z_3$

In the rest of the thesis, we will depict derivations as above. Here is a graphical representation of $F^{\mathcal{D}}$, where r is represented with a circle and p with arrows:



Notice that the last two triggers do not produce any atoms. At the fourth element $D_3 = (\mathfrak{t}_3, F_3, Z_3)$ of \mathcal{D} we have that the union of all the transitory factbases produced by the previous active factbase is simply $F_3 = \{r(a), p(a, y_{\mathfrak{t}_1}), p(y_{\mathfrak{t}_1}, a), p(a, a)\}$ (since the last time the active factbase changed is on Z_2). The substitution $\sigma = \{y_{\mathfrak{t}_1} \mapsto a\}$ is a retraction from F_3 to $\{r(a), p(a, a)\}$. By applying σ we therefore arrive at the final active factbase Z_3 which is in fact equal to the initial factbase F plus the loop $p(a, a)$. ■

After the application of \mathfrak{t}_3 in the above derivation, we arrive almost at the same factbase that we began with. One could imagine an oscillatory situation, where we reapply the rules to this new-old active factbase producing the same atoms over and over. This is prevented by the condition that every trigger is new when introduced, i.e. it appears at most once in a derivation. In this case there are no more triggers to apply after \mathfrak{t}_5 and the derivation necessarily stops at this point.

On the other hand, it is important to point out that a derivation does not need to satisfy any concept of completeness, i.e. it does not necessarily include

all possible rule applications. To this end, we will introduce the concepts of *exhaustivity* and *termination* with respect to a class of derivations at the end of this section. As a limit case, we will call the derivation with only one element $D_0 = (\emptyset, F, F)$, the *trivial* derivation from (F, \mathcal{R}) (where \mathcal{R} is any ruleset). Below is another example of a derivation:

Example “GENERIC” 8 (continued from Examples 3 and 4): We remind that $F = \{p(a, b), p(c, d), r(e)\}$ and \mathcal{R} is the following set of rules:

$$R_1 = p(x, y) \wedge r(z) \rightarrow p(y, z)$$

$$R_2 = p(x, y) \wedge p(y, z) \rightarrow \exists u p(z, u)$$

$$R_3 = p(x, y) \wedge p(x, z) \rightarrow p(y, z)$$

Here is a derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	$F_0 = F$	$Z_0 = F_0$
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b, z \mapsto e\})$	$F_1 = F \cup \{p(b, e)\}$	$Z_1 = F_1$
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto b, z \mapsto e\})$	$F_2 = F_1 \cup \{p(e, u_{\mathbf{t}_2})\}$	$Z_2 = F_2$
$\mathbf{t}_3 = (R_2, \{x \mapsto b, y \mapsto e, z \mapsto u_{\mathbf{t}_2}\})$	$F_3 = F_2 \cup \{p(u_{\mathbf{t}_2}, u_{\mathbf{t}_3})\}$	$Z_3 = F_3$
$\mathbf{t}_4 = (R_1, \{x \mapsto c, y \mapsto d, z \mapsto e\})$	$F_4 = F_3 \cup \{p(d, e)\}$	$Z_4 = F_4$

Notice that \mathcal{D} can be extended with the addition of more derivation triples. Below we find a representation of $F^{\mathcal{D}}$, where atoms in F are colored blue and subsequent dependencies are represented by a change of color. This is the concept of *rank*, which we will soon introduce.

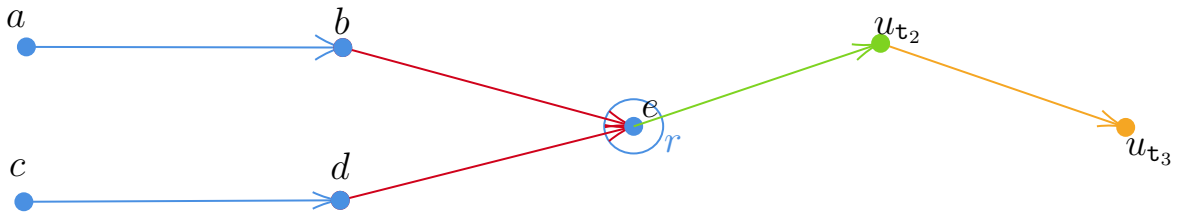


Figure 4.2: Example “GENERIC”, derivation \mathcal{D} : representation of $F^{\mathcal{D}}$.

Notice also that in every derivation triple D_i of \mathcal{D} the active factbases Z_i are always equal to the transitory factbases F_i . In other words when an atom is produced, it remains permanently in the (active) factbase. ■

Definition 4.2. A derivation $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ is *monotonic* if for every i holds that $Z_i = F_i$. —

The characterization of such a derivation as monotonic is due to the fact that the (transitory) factbase is growing monotonously after each rule application, i.e. $F_i \subseteq F_{i+1}$ for every i .⁴ The concept of monotonic derivation as is presented here, despite underlining the role of triggers, agrees with the typical paradigm of “derivation” in the related literature [1, 55, 8, 56]. However, with the term “derivation” in this thesis we are specifying a broader notion in an attempt to define a framework general enough to model many different forward chaining scenarios. It is easy to verify that from any derivation \mathcal{D} , we can construct a monotonic derivation \mathcal{D}' such that $\text{trig}(\mathcal{D}) = \text{trig}(\mathcal{D}')$, in the following manner:

Remark 4.1. Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ be a derivation. Then $\mathcal{D}' = (\mathfrak{t}'_*, F'_*, Z'_*)$ where for all i holds that $\mathfrak{t}'_i = \mathfrak{t}_i$, $F'_0 = F_0$, $F'_i = F'_{i-1} \cup F_i$ and $Z'_i = F'_i$, is a monotonic derivation. ♣

It is important to underline that there is no relation between non-monotonic derivations and non-monotonic logic. That is because all the information that is removed from the factbase is redundant, therefore the active factbase is semantically equivalent with the set of produced atoms. Indeed, the condition ii) of the definition of derivation, assures that the union of all the factbases produced from the previous active factbase (Z_ℓ) is logically equivalent to the new active factbase (Z_i).

Observing the derivation \mathcal{D} from the previous example, we can see that trigger \mathfrak{t}_4 is applicable directly on the initial factbase F , and does not require any other rule application to precede it. In this case we say that \mathfrak{t}_4 is of *rank* 1, as it is among the triggers that are applicable on the initial factbase. The concept of rank applies to triggers, atoms and derivation triples. Below we specify this with a recursive definition:

⁴In Subsection 4.3.3 we will discuss the concept of *submonotonic* derivations, which characterizes the monotonicity in the sequence of active factbases.

Definition 4.3 (Rank, Depth). Given a derivation $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$, we define the *rank* of an atom $A \in F^\mathcal{D}$ as follows: $rank_\mathcal{D}(A) = 0$ if $A \in F_0$, otherwise if \mathfrak{t}_i produces A in \mathcal{D} , then $rank_\mathcal{D}(A) = 1 + \max\{rank_\mathcal{D}(A') \mid A' \in \mathbf{sp}(\mathfrak{t}_i)\}$. This concept is naturally extended to triggers as well as to derivation triples:

- $rank_\mathcal{D}(\mathfrak{t}) = 1 + \max\{rank_\mathcal{D}(A') \mid A' \in \mathbf{sp}(\mathfrak{t})\}$,
- $rank_\mathcal{D}(D_0) = 0$ and $rank_\mathcal{D}(D_i) = rank_\mathcal{D}(\mathfrak{t}_i)$ for $i > 0$.

Moreover, for every $t \in \mathbf{term}(F^\mathcal{D})$ we define:

- $rank_\mathcal{D}(t) = \min\{rank_\mathcal{D}(A) \mid A \in F^\mathcal{D}, t \in \mathbf{term}(A)\}$.

The *depth* of a finite derivation is the maximal rank of all atoms that are produced in it. \dashv

Notice that if $\mathfrak{t} \notin \mathbf{trig}(\mathcal{D})$ but $\mathbf{sp}(\mathfrak{t}) \subseteq F^\mathcal{D}$, then the above formula still serves as a definition for the rank of \mathfrak{t} in \mathcal{D} , i.e. triggers that do not appear in \mathcal{D} , but whose support is inferred with \mathcal{D} have well defined ranks in \mathcal{D} . When \mathcal{D} is implied by the context, we will simply write $rank(\cdot)$ instead of $rank_\mathcal{D}(\cdot)$. An important class of derivations are those where the elements are ordered according to rank:

Definition 4.4. A derivation $\mathcal{D} = D_0, D_1, \dots$ is *rank compatible* if for all elements D_i and D_j in \mathcal{D} with $i < j$, the rank of the trigger of D_i is smaller or equal to the rank of the trigger of D_j . Furthermore, in a rank compatible derivation, every element D_i which is the final of its rank (i.e. for every other element D_j of \mathcal{D} , if $rank_\mathcal{D}(D_j) \leq rank_\mathcal{D}(D_i)$ then $j < i$), is called a *rank mark*. \dashv

In the two examples of derivations that we have presented, we already have one rank compatible derivation (Example 7) and one that is not rank compatible (Example 3.8). Anyhow, in all the following figures, ranks will be represented with the same colors, namely blue, red, green, orange, brown and yellow for ranks 0 to 5 respectively. In addition, in our tabular representations of rank compatible derivations, the changes of ranks will be illustrated with thick blue horizontal lines and we will add a fourth column where the rank of each rank mark will be indicated (starting from Example 9 in the following section).

An interesting situation appears when a trigger t in a derivation \mathcal{D} does not produce any atom. This happens when the corresponding rule is a datalog rule, and all atoms of the output of t are already produced in the current prefix of \mathcal{D} (for instance look at the triggers t_4 and t_5 in the Example 7). But it is easy to observe that whenever a trigger does produce at least one atom, then its rank is equal to the rank of the atoms that it produces. Finally, we remark that it is impossible for two rank marks to be of the same rank in a derivation.

We can see how a certain class of derivations can be modeled with the α operator (as introduced in Subsection 2.2.2), but not all. The definition of a derivation is purposefully general, in order to be able to represent many possible forward chaining scenarios. Given a knowledge base (F, \mathcal{R}) , we want to derive information with forward chaining in the most effective way (where “effective” depends on context and applications). In the next sections we will demonstrate several strategies which have different features. The abstract notion of a *chase variant* corresponds to a process that builds derivations of a certain kind, usually with the aim of obtaining a universal model of the knowledge base. Below is a formal definition:

Definition 4.5. A *chase variant* is a class of derivations. Given a chase variant X , each derivation that belongs to X will be called an *X-derivation*. In addition, if $\mathcal{D} = D_0, D_1, \dots, D_n$ is an X -derivation, and there exists an X -derivation $\mathcal{D}' = D_0, D_1, \dots, D_n, D_{n+1}$ with $D_{n+1} = (t, F_{n+1}, Z_{n+1})$, then the trigger t is said to be *X-applicable* on \mathcal{D} . \dashv

Therefore in a formal sense, a chase variant is simply a family (class) of derivations that can be specified by imposing any kind of restrictions, for example on which triggers can be applied and when can they be applied, or which kind of retractions will produce the active factbases⁵. In the following sections we will see several examples of chase variants, but here we first define some basic notions concerning chase variants. The most important are those of *termination* and *exhaustivity*.

⁵In this thesis we did not need to explicitly demand that every chase variant X is *prefix-closed*, i.e. if $\mathcal{D} \in X$, then every prefix \mathcal{D}' of \mathcal{D} is also in X . This feature can be added without major consequences, if we broaden our definition of breadth-first variants by including also all their prefixes (see Definition 4.8).

Definition 4.6. An X-derivation $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ from (F, \mathcal{R}) is *terminating* if it is finite and there does not exist a trigger \mathfrak{t} that is X-applicable on \mathcal{D} . \dashv

X-applicability is defined on finite derivations. Can we conceptualize an infinite derivation where no trigger is X-applicable? Exhaustivity is the property that no trigger is indefinitely delayed: if a trigger is X-applicable on some prefix of the X-derivation, then either it is applied at some later point, or it becomes non-X-applicable at some later point in the derivation. This property applied on a finite X-derivation leads to termination, since it implies that no triggers are X-applicable to the derivation⁶. The formal definition follows:

Definition 4.7. An X-derivation $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ from (F, \mathcal{R}) is *exhaustive* if for all prefixes $\mathcal{D}' = D_1, \dots, D_i$ of \mathcal{D} , if a trigger \mathfrak{t} from \mathcal{R} is X-applicable on \mathcal{D}' , then there exists a $k > i$ such that one of the following two holds:

1. $\mathfrak{t}_k = \mathfrak{t}$ or
2. \mathfrak{t} is not X-applicable on D_1, \dots, D_k . \dashv

Therefore according to the above definitions, an X-derivation is terminating if and only if it is finite and exhaustive. Exhaustivity is also known as *fairness* in the literature [57, 20].

There can possibly be many very different X-derivations from a particular knowledge base. For every chase variant X presented in this thesis, a terminating X-derivation from a knowledge base (F, \mathcal{R}) produces as the last active factbase a universal model of (F, \mathcal{R}) [37]. So the question that naturally rises is whether there are any effective strategies in order to find terminating X-derivations.

A reasonable approach is to first apply the triggers whose support is included in the initial (active) factbase, then the triggers whose support is at most or rank 1 and so on. It is important to notice that rank compatibility alone does not ensure that all possible rule applications are made before proceeding to the following rank. This is the case in a *breadth-first* scenario. The breadth-first paradigm of forward chaining was introduced using the α operator in Section 2.2.

⁶taking into account that a finite derivation is also a prefix of itself.

Definition 4.8 (Breadth-first). Let X be a chase variant. An X -derivation $\mathcal{D} = D_0, D_1, \dots$ is *breadth-first* if it is rank compatible and for every rank mark D_i , there does not exist an X -derivation $\mathcal{D}' = D_0, D_1, \dots, D_i, D'_{i+1}$ with $\text{rank}(D'_{i+1}) = \text{rank}(D_i)$. If every X -derivation is breadth-first then X is called a *breadth-first* chase variant. \dashv

The breadth-first approach is the most popular strategy of choosing how to proceed in a derivation. In Section 4.4 we will examine its efficiency in different circumstances (i.e. for different chase variants). Before moving on to present several chase variants, we emphasize a property of finite non-terminating breadth-first X -derivations: every trigger that is X -applicable on such a derivation will be of rank strictly higher than all the triggers associated with it.

Lemma 4.1. Let $\mathcal{D} = D_0, D_1, \dots, D_n$ be a finite breadth-first X -derivation. If \mathcal{D} is not terminating, then every X -derivation $\mathcal{D}' = D_0, D_1, \dots, D_n, D'_{n+1}$ has the property that $\text{rank}(D'_{n+1}) > \text{rank}(D_n)$. \clubsuit

As a final observation regarding the notions introduced above, we highlight that assuming X_1 and X_2 are two chase variants, a derivation \mathcal{D} can be at the same time an X_1 -derivation as well as an X_2 -derivation. Even more, \mathcal{D} can be exhaustive or terminating or breadth-first with respect to X_1 , but not exhaustive or not terminating or not breadth-first with respect to X_2 . In the following sections we will see such examples.

4.2 Monotonic Chase Variants

A chase variant is usually characterized/defined by a general property that contributes to avoiding some of the possible redundancies in the resulting factbase. Below we find a simple case where the redundancy is evident. It is also the first of our examples where an extra column and thick blue lines are added to the table representing a rank compatible derivation, indicating the corresponding ranks:

Example 9: Suppose we have the rule $R = p(x, y) \rightarrow \exists z p(x, z) \wedge q(z)$ and let $F = \{p(a, b)\}$. Here is an n -long derivation from $(F, \{R\})$:

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathbf{t}_1 = (R, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{p(a, z_{\mathbf{t}_1}), q(z_{\mathbf{t}_1})\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{p(a, z_{\mathbf{t}_2}), q(z_{\mathbf{t}_2})\}$	$Z_2 = F_2$	2
$\mathbf{t}_3 = (R, \{x \mapsto a, y \mapsto z_{\mathbf{t}_2}\})$	$F_3 = F_2 \cup \{p(a, z_{\mathbf{t}_3}), q(z_{\mathbf{t}_3})\}$	$Z_3 = F_3$	3
\vdots	\vdots	\vdots	
$\mathbf{t}_n = (R, \{x \mapsto a, y \mapsto z_{\mathbf{t}_{n-1}}\})$	$F_n = F_{n-1} \cup \{p(a, z_{\mathbf{t}_n}), q(z_{\mathbf{t}_n})\}$	$Z_n = F_n$	n

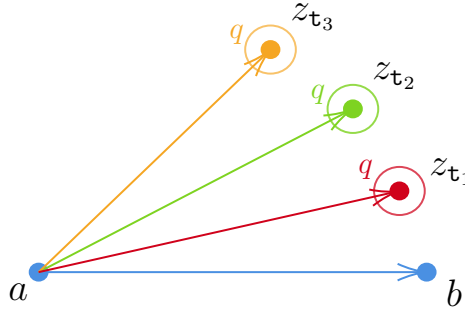


Figure 4.3: Example 9, Evolving factbase.

In Figure 4.3 we see the evolution of the factbase. We can see how a derivation in this style can be infinite, whereas after the first rule application, all the others do not contribute to anything new semantically, i.e $F_1 \models F_i$ for all $i \leq n$. ■

The derivation of the above example belongs to the *oblivious chase* variant, which corresponds with our definition of monotonic derivation. The obvious way to tackle the problem illustrated in this example, is to impose a condition of the form $F_{i-1} \not\models F_i$ for all i , and then we would have a derivation where definitely every rule application would add “semantic value”. This is indeed the case in a chase variant called *equivalent*, which we will define in the next paragraph. However it can be computationally costly to test for logical implication of entire factbases at every step⁷ and there are other filters which can deal with certain types of redundancies in a more effective way. For example in the above case, we notice that the mapping of the non-frontier variable y has no (semantic) implication on the added information when we apply the rule. Therefore we can say that we do not need to repeat triggers which share the

⁷testing for graph homomorphism is NP-complete.

same rule and the same mapping of frontier variables to the factbase. This is what the *semi-oblivious* chase does.

We will present four monotonic chase variants, namely oblivious (**O**), semi-oblivious (**SO**), restricted (**R**), equivalent chase (**E**). All the monotonic chase variants are (monotonic) derivations that comply with some condition of applicability of the triggers.

Definition 4.9 (**O**-, **SO**-, **R**- and **E**-applicability). Let \mathcal{D} be a derivation from a knowledge base (F, \mathcal{R}) . A trigger τ is called:

1. **O-applicable** on \mathcal{D} if τ is applicable on $F^{\mathcal{D}}$ and $\tau \notin \text{trig}(\mathcal{D})$.
2. **SO-applicable** on \mathcal{D} if $\tau = (R, \pi)$ is applicable on $F^{\mathcal{D}}$ and for every trigger $\tau' = (R, \pi')$ in the sequence of triggers associated with \mathcal{D} , the restrictions of π and π' to the frontier of R are not equal.
3. **R-applicable** on \mathcal{D} if τ is applicable on $F^{\mathcal{D}}$ and there does not exist a retraction from $\text{op}(\tau) \cup F^{\mathcal{D}}$ to $F^{\mathcal{D}}$.
4. **E-applicable** on \mathcal{D} if
 - (i) τ is applicable on $F^{\mathcal{D}}$ and there does not exist a homomorphism from $F^{\mathcal{D}} \cup \text{op}(\tau)$ to $F^{\mathcal{D}}$, i.e. it holds that $F^{\mathcal{D}} \not\cong F^{\mathcal{D}} \cup \text{op}(\tau)$.
 - (ii) for every τ' that also satisfies property (i), it holds that $\text{rank}_{\mathcal{D}}(\tau) \leq \text{rank}_{\mathcal{D}}(\tau')$. \dashv

Notice that when $X \in \{\mathbf{O}, \mathbf{SO}, \mathbf{E}\}$, the X -applicability of the trigger depends not only on the final factbase $F^{\mathcal{D}}$, but also on the whole derivation \mathcal{D} . This is not the case for **R**-applicability. This motivates the following definition:

Definition 4.10 (**R**-applicability on Factbase). A trigger τ is **R-applicable** on a factbase F if τ is applicable on F and F is not a retract of $F \cup \text{op}(\tau)$. \dashv

The definitions of the monotonic chase variants follow directly the conditions of applicability that we outlined above.

Definition 4.11 (Monotonic Chase Variants). Let F be a factbase and \mathcal{R} be a ruleset. We define four monotonic chase variants:

- I. An **oblivious derivation** is any monotonic derivation \mathcal{D} from (F, \mathcal{R}) .
- II. A **semi-oblivious derivation** is any monotonic derivation \mathcal{D} from (F, \mathcal{R}) such that for every element $D_i = (\mathfrak{t}_i, F_i, Z_i)$ of \mathcal{D} , the trigger \mathfrak{t}_i is SO-applicable on the prefix D_0, D_1, \dots, D_{i-1} of \mathcal{D} .
- III. A **restricted derivation** is any monotonic derivation \mathcal{D} from (F, \mathcal{R}) such that for every element $D_i = (\mathfrak{t}_i, F_i, Z_i)$ of \mathcal{D} , the trigger \mathfrak{t}_i is **R**-applicable on the prefix D_0, D_1, \dots, D_{i-1} of \mathcal{D} .
- IV. An **equivalent derivation** is any monotonic derivation \mathcal{D} from (F, \mathcal{R}) such that for every element $(\mathfrak{t}_i, F_i, Z_i)$ of \mathcal{D} , the trigger \mathfrak{t}_i is **E**-applicable on the prefix D_0, D_1, \dots, D_{i-1} of \mathcal{D} . ⊢

We will abbreviate the above types of derivations with **O**-derivation, **SO**-derivation, **R**-derivation, and **E**-derivation, respectively. Furthermore, the corresponding classes of derivations, i.e. chase variants, will be called **O**-chase, **SO**-chase, **R**-chase, and **E**-chase, respectively. In the following section, we will present more chase variants.

Proposition 4.1 (Monotonic Chase Hierarchy [37]). The following containment relation holds between the four monotonic chase variants:
 $\mathbf{E} \subset \mathbf{R} \subset \mathbf{SO} \subset \mathbf{O}$. ♣

We now describe those chase variants in more detail. The semi-oblivious chase [37, 14] was initially defined as a reformulation of the *skolem* chase [16]. The skolem chase consists of first skolemizing the rules (by replacing existentially quantified variables with skolem functions whose arguments are the frontier variables) then running the oblivious chase. This procedure yields isomorphic results with the **SO**-chase as defined above, in the sense that both generate exactly the same sets of atoms, up to a bijective renaming of the new

variables by skolem terms. The SO-chase, although it is an evident optimization over the O-chase, is not very potent in filtering out redundancy, even in some seemingly simple cases as the following:

Example 10: Let $R = p(x, y) \rightarrow \exists z p(y, z) \wedge p(z, x)$ and $F = \{p(a, b)\}$. Let \mathcal{D} be the following derivation of length n from $(F, \{R\})$:

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathfrak{t}_1 = (R, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{p(b, z_{\mathfrak{t}_1}), p(z_{\mathfrak{t}_1}, a)\}$	$Z_1 = F_1$	1
$\mathfrak{t}_2 = (R, \{x \mapsto b, y \mapsto z_{\mathfrak{t}_1}\})$	$F_2 = F_1 \cup \{p(z_{\mathfrak{t}_1}, z_{\mathfrak{t}_2}), p(z_{\mathfrak{t}_2}, b)\}$	$Z_2 = F_2$	2
$\mathfrak{t}_3 = (R, \{x \mapsto z_{\mathfrak{t}_1}, y \mapsto z_{\mathfrak{t}_2}\})$	$F_3 = F_2 \cup \{p(z_{\mathfrak{t}_2}, z_{\mathfrak{t}_3}), p(z_{\mathfrak{t}_3}, z_{\mathfrak{t}_1})\}$	$Z_3 = F_3$	3
\vdots	\vdots	\vdots	\vdots
$\mathfrak{t}_n = (R, \{x \mapsto z_{\mathfrak{t}_{n-2}}, y \mapsto z_{\mathfrak{t}_{n-1}}\})$	$F_n = F_{n-1} \cup \{p(z_{\mathfrak{t}_{n-1}}, z_{\mathfrak{t}_n}), p(z_{\mathfrak{t}_n}, z_{\mathfrak{t}_{n-2}})\}$	$Z_n = F_n$	n

In Figure 4.4 we have a representation of the evolution of the factbase of \mathcal{D} . Since the mapping of the frontier variables changes at every element, \mathcal{D} is a valid SO-derivation. However already after the first rule application, we stop to entail new information, i.e. $F_1 \models F_i$ for every $i \leq n$. The

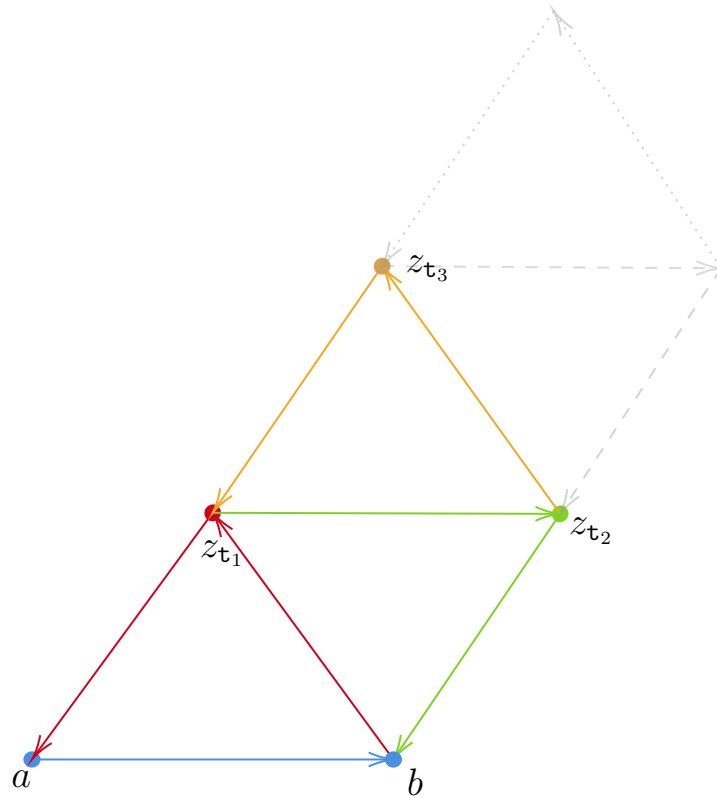


Figure 4.4: Example 10, evolving factbase.

SO-oblivious chase cannot detect this redundancy. The **R**-chase in contrast does detect it, as for the second rule application there is an extension of $\{x \mapsto b, y \mapsto z_{t_1}\}$, namely $\{x \mapsto b, y \mapsto z_{t_1}, z_{t_2} \mapsto a\}$, which if applied to the head of R (i.e. $\{p(y, z), p(z, x)\}$), results in a subset of F_1 . Therefore \mathcal{D} is not a **R**-derivation. In fact the only **R**-derivation possible from $(F, \{R\})$ amounts to just performing the first rule application of \mathcal{D} and then halting. The **E**-chase will behave similarly in this case. ■

The restricted chase, which is also known as the *standard* chase, provides a reasonable local redundancy check, where we verify to see if the new facts resulting from a possible immediate derivation, are indeed *new*, meaning that the new variables cannot be mapped back to terms of the current factbase with a retraction. This is a plausible and relatively easily verifiable condition, which stems from the perspective that rules are constraints, and repairing databases with constraints is commonly done at a local level.

The equivalent chase [55] was initially conceived as a monotonic way to simulate the *core chase* [18] with regard to termination. The core chase, which will be formally presented in the following section, is a chase algorithm that calculates the finite universal model of the knowledge base if and only if it exists. It proceeds in a breadth-first manner by performing in parallel all rule applications according to **R**-applicability and then it computes a core of the resulting factbase. We remind that a core of a set of atoms is one of its minimal equivalent subsets. Hence, the core chase may at some step remove atoms that were introduced at a former step. In a monotone setting we cannot retain the core at each step, but we can instead test for logical equivalence. This follows from the fact that $F_i \equiv F_{i+1}$ if and only if $\text{core}(F_i)$ is isomorphic to $\text{core}(F_{i+1})$ (up to bijective variable renaming). In order to be in accordance with the core chase, the equivalent chase is designed to run in a breadth-first manner, which is why **E**-applicability is satisfied only by triggers of minimal rank (as specified in Definition 4.9). Indeed, each exhaustive **E**-derivation is a breadth-first **E**-derivation. However, the higher rank of an **E**-derivation might be incomplete, i.e. some **E**-applicable triggers from this rank might be missing. This is not allowed by our definition of breadth-first **X**-derivation. Hence we

can say that:

Remark 4.2. Every **E**-derivation is a prefix of a breadth-first **E**-derivation. ♣

Considering this remark, we will generally regard the **E**-chase as a breadth-first chase variant (even though technically it includes also the prefixes of breadth-first **E**-derivations). Below is an example which serves to show how there are equivalent chase derivations that terminate on knowledge bases where every restricted chase derivation does not terminate.

Example “EQUICORE” 11: We have that $F = \{p(a, b)\}$ and \mathcal{R} the following set of rules:

$$R_1 = p(x, y) \rightarrow \exists z p(y, z)$$

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

Below is the **E**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{p(b, z_{\mathbf{t}_1})\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto b, z \mapsto z_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{p(b, b)\}$	$Z_2 = F_2$	2

Any trigger from \mathcal{R} applied on Z_2 results in a factbase that is semantically equivalent with Z_2 . Hence \mathcal{D} is terminating as an **E**-derivation.

Besides, \mathcal{D} is also a **R**-derivation. But it is not a terminating **R**-derivation. Indeed, the trigger $\mathbf{t}_3 = (R_1, \{x \mapsto b, y \mapsto z_{\mathbf{t}_1}\})$ is **R**-applicable on Z_2 . Actually there does not exist a terminating **R**-derivation from (F, \mathcal{R}) , because starting from (F, \mathcal{R}) , the loop $p(z_\star, z_\star)$ is never added to the end of the p -path, so R_1 will be **R**-applicable on any resulting factbase. ■

As will be discussed in detail in Section 4.4, for $X \in \{\mathbf{O}, \mathbf{SO}, \mathbf{E}\}$, if there exists a terminating **X**-derivation for a given knowledge base, then all exhaustive **X**-derivations from this knowledge base are terminating. That holds because the order in which rules are applied does not affect the detection (or not) of redundancies. On the other hand the restricted chase does not behave well in certain cases, as it has an overall much more non-deterministic nature in comparison to the rest of the monotonic chase variants. Indeed, it can produce very different derivations from the same knowledge base. Here is an example:

Example 12: We have two rules, $R_1 = p(x, y) \rightarrow \exists z p(y, z)$ and $R_2 = p(x, y) \rightarrow p(y, y)$ and the factbase is $F = \{p(a, b)\}$. Let $\pi = \{x \mapsto a, y \mapsto b\}$. Then the triggers $\tau_1 = (R_1, \pi)$ and $\tau_2 = (R_2, \pi)$ are both **R**-applicable on F . If τ_2 is applied first, then none of the rules are applicable to the resulting factbase, which is $\{p(a, b), p(b, b)\}$, yielding a terminating **R**-derivation. However if we apply τ_1 first and τ_2 second, we produce the factbase $F_2 = \{p(a, b), p(b, z_{\tau_1}), p(b, b)\}$ and with $\pi' = \{x \mapsto b, y \mapsto z_{\tau_1}\}$ we have that $\tau'_1 = (R_1, \pi')$ as well as $\tau'_2 = (R_2, \pi')$ are again both **R**-applicable. Consequently, if we always choose to apply R_1 before R_2 then the corresponding derivation will be infinite (and exhaustive). ■

4.3 Non-monotonic Chase Variants

The main choice when specifying a monotonic chase variant revolves around the applicability condition. However the degrees of liberty increase drastically when we have the freedom to choose any way of forming our active factbases, as long as it is the result of a retraction from the union of the transitory factbases produced by the previous active factbase (in compliance with Definition 4.1). Hence, there is a major shift in viewpoint from monotonic to non-monotonic forward chaining, with the active factbase here being of principal importance in defining a chase variant. Indeed in all the chase variants that we will present in this section, namely the *vacuum* (V), the *frugal* (F), the *parallel* (P) and the *core* (C) *chase*, a trigger is applicable on a derivation if it is **R**-applicable on the final active factbase. Their variation between them comes from their behavior towards the active factbase. In the vacuum and the frugal chase, the application of a trigger can cause several redundant atoms to be removed from the active factbase. In the parallel chase, the active factbase expands only when all triggers of the current rank have been applied. The core chase uses the same mechanism as the parallel chase but only keeps a core as the new active factbase.

4.3.1 Vacuum Chase & Frugal Chase

The vacuum chase, which is introduced here for the first time, is inspired by the frugal chase [28, 58, 59]. Those two chase variants are very similar, hence will be presented simultaneously. They add an interesting perspective to the whole chase landscape, by designating a sophisticated mechanism to eliminate more redundancy than the restricted chase. In Subsection 2.1.3 we introduced the notion of *piece* (Definition 2.5). The vacuum chase is based on the observation that there are times when it is easy to verify that the output of a specific trigger might render a piece in the current factbase redundant. Using this observation we can improve the **R**-chase, which sometimes produces infinite derivations based exactly on those kinds of redundant pieces.

Example “FRUGALPHA” 13: Let $F = \{r(a)\}$ and \mathcal{R} is the following set of rules:

$$R_1 = r(x) \rightarrow \exists z p(x, z)$$

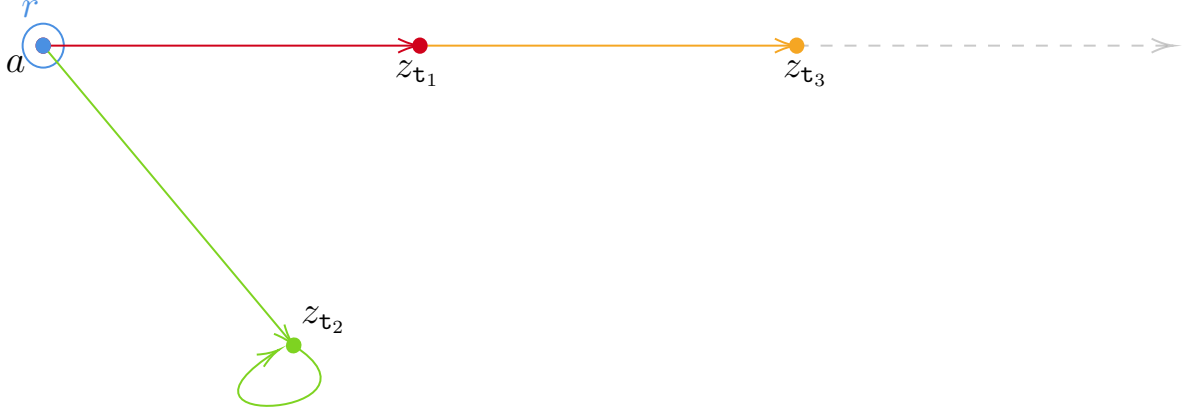
$$R_2 = p(x, y) \rightarrow \exists z p(x, z) \wedge p(z, z)$$

$$R_3 = p(x, y) \rightarrow \exists z p(y, z)$$

Here is a **R**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$t_1 = (R_1, \{x \mapsto a\})$	$F_1 = F \cup \{p(a, z_{t_1})\}$	$Z_1 = F_1$	1
$t_2 = (R_2, \{x \mapsto a, y \mapsto z_{t_1}\})$	$F_2 = F_1 \cup \{p(a, z_{t_2}), p(z_{t_2}, z_{t_2})\}$	$Z_2 = F_2$	
$t_3 = (R_3, \{x \mapsto a, y \mapsto z_{t_1}\})$	$F_3 = F_2 \cup \{p(z_{t_1}, z_{t_3})\}$	$Z_3 = F_3$	2
$t_4 = (R_3, \{x \mapsto z_{t_1}, y \mapsto z_{t_3}\})$	$F_4 = F_3 \cup \{p(z_{t_3}, z_{t_4})\}$	$Z_4 = F_4$	3
\vdots	\vdots	\vdots	
$t_n = (R_3, \{x \mapsto z_{t_{n-2}}, y \mapsto z_{t_{n-1}}\})$	$F_n = F_{n-1} \cup \{p(z_{t_{n-1}}, z_{t_n})\}$	$Z_n = F_n$	$n-1$

Below is a representation of the (evolution of the) active factbase:



The above **R**-derivation can grow infinitely, however none of the applications of rule R_3 adds anything new to our factbase. Indeed, for all $i \leq n$, $F_2 \models F_i$. Lastly, note that by adding an application of R_2 between every two applications of R_3 we can also create an infinite exhaustive **R**-derivation from (F, \mathcal{R}) . ■

Following the derivation of the above example, we can see how, when t_2 is applied, the atom $p(a, z_{t_1})$ becomes redundant and moreover it holds that $\text{op}(t_2)$ is a retract of $\text{op}(t_2) \cup \{p(a, z_{t_1})\}$. But the whole chain of rule applications that results in the non-termination of a derivation like \mathcal{D} , is based on the existence (and “survival”) of this atom after the application of t_2 . Therefore what we can do is to scan for possible pieces of the factbase that are homomorphic to a subset of the output of our current trigger, and remove them from the resulting factbase. The vacuum chase does this and even more: it also divides the output of the trigger to pieces, and adds only those which are necessary to be added.

Here we introduce a few notions that are needed to specify the vacuum and the frugal chase. Let $t = (R, \pi)$ be a trigger. An *output piece* of t is a minimal non-empty subset $\overline{\text{op}}(t) \subseteq \text{op}(t)$ with the property that if $A \in \overline{\text{op}}(t)$, then for every $A' \in \text{op}(t)$, if the atoms A and A' have at least one common new variable, i.e. if $\text{nul}(\text{op}(t)) \cap \text{var}(A) \cap \text{var}(A') \neq \emptyset$, then it holds that $A' \in \overline{\text{op}}(t)$.

Example “FRUGBETA” 14: Let $R = p(x, y) \rightarrow \exists z \exists w p(z, x) \wedge p(y, w)$, $\pi = \{x \mapsto a, y \mapsto b\}$ and $t = (R, \pi)$. Then the pieces in the head of R are $\{p(z, x)\}$ and $\{p(y, w)\}$, whereas the pieces in $\text{op}(t)$ are $\{p(z_t, a)\}$ and $\{p(b, w_t)\}$. ■

Now suppose that $\{\text{op}_1(\mathfrak{t}), \dots, \text{op}_n(\mathfrak{t})\}$ are the output pieces of a trigger \mathfrak{t} and let F be a factbase. The *frugal output* $\text{frop}(\mathfrak{t}, F)$ of \mathfrak{t} with respect to F is the union of all the pieces $\text{op}_i(\mathfrak{t}) \subseteq \text{op}(\mathfrak{t})$ which have the property that F is not a retract of $F \cup \text{op}_i(\mathfrak{t})$.

Example “FRUGBETA” 15 (continued from Example 14):

Let $F = \{p(a, b), p(b, c)\}$. We can see that F is a retract of $F \cup \{p(b, w_{\mathfrak{t}})\}$ (by mapping $w_{\mathfrak{t}}$ to c), whereas $F \cup \{p(z_{\mathfrak{t}}, a)\}$ is a core. Therefore the frugal output of \mathfrak{t} with respect to F is $\text{frop}(\mathfrak{t}, F) = \{p(z_{\mathfrak{t}}, a)\}$. ■

Finally, suppose that $Z \subseteq F^{\mathcal{D}}$, where \mathcal{D} is a derivation. A piece P in Z is *subsumed* by $\text{frop}(\mathfrak{t}, Z)$ if there is a retraction σ from $\text{frop}(\mathfrak{t}, Z) \cup P$ to $\text{frop}(\mathfrak{t}, F)$. A piece P in Z is *isomorphically subsumed* by $\text{frop}(\mathfrak{t}, Z)$ if there is an isomorphism σ from P to a subset of $\text{frop}(\mathfrak{t}, Z)$ such that σ is also a retraction from $\text{frop}(\mathfrak{t}, Z) \cup P$ to $\text{frop}(\mathfrak{t}, F)$.

Example “FRUGBETA” 16 (continued from Examples 14 and 15): Suppose that $Z = F \cup \{p(z_{\mathfrak{t}}, a)\}$. Since only one atom has a variable in Z , each atom is a piece in Z . Let $R' = p(x, y) \rightarrow \exists z p(z, x) \wedge s(z)$ and $\mathfrak{t}' = (R', \pi)$. Then we have only one piece in $\text{op}(\mathfrak{t}')$ (itself). And $\text{frop}(\mathfrak{t}', Z) = \text{op}(\mathfrak{t}')$. Then, by mapping $z_{\mathfrak{t}}$ to $z_{\mathfrak{t}'}$ we can see that the piece $\{p(z_{\mathfrak{t}}, a)\}$ of Z is subsumed by $\text{frop}(\mathfrak{t}', Z)$. ■

The vacuum chase removes all the pieces that are subsumed by the frugal output of the current trigger \mathfrak{t} with respect to the current active factbase Z , while the frugal chase removes all the pieces that are isomorphically subsumed by the frugal output of \mathfrak{t} with respect to Z . In particular, we denote with:

- $\mathbf{V}(\mathfrak{t}, Z)$ the union of all the pieces in Z that are not subsumed by $\text{frop}(\mathfrak{t}, Z)$.
- $\mathbf{F}(\mathfrak{t}, Z)$ the union of all the pieces in Z that are not isomorphically subsumed by $\text{frop}(\mathfrak{t}, Z)$.

We are now ready to provide the definitions of the vacuum and the frugal chase:

Definition 4.12 (Vacuum & Frugal Chase). Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ be a derivation from (F, \mathcal{R}) . \mathcal{D} is a **vacuum derivation** if for all $i > 0$ holds that

- the trigger \mathfrak{t}_i is **R**-applicable to Z_{i-1} and
- $Z_i = \mathbf{V}(\mathfrak{t}_i, Z_{i-1}) \cup \mathbf{frop}(\mathfrak{t}_i, Z_{i-1})$.

\mathcal{D} is a **frugal derivation** if for all $i > 0$ holds that

- the trigger \mathfrak{t}_i is **R**-applicable to Z_{i-1} and
- $Z_i = \mathbf{F}(\mathfrak{t}_i, Z_{i-1}) \cup \mathbf{frop}(\mathfrak{t}_i, Z_{i-1})$. ⊥

By definition, for every Z and \mathfrak{t} holds that $\mathbf{V}(\mathfrak{t}, Z) \subseteq \mathbf{F}(\mathfrak{t}, Z)$, i.e. the vacuum chase removes from the active factbase all the redundant atoms that the frugal chase removes, and even more. But from an algorithmic viewpoint, it does not seem that calculating $\mathbf{V}(\mathfrak{t}, Z)$ is on average going to be much harder than calculating $\mathbf{F}(\mathfrak{t}, Z)$.⁸ So then why choose $\mathbf{F}(\mathfrak{t}, Z)$ and restrict the removal of redundant atoms? The initial specification of the frugal chase [28, 59], although equivalent with the one presented above, assumes another operational perspective, from where the progression from frugal chase to vacuum chase is not at all evident. In the appendix, Section (D), we elaborate on this point.

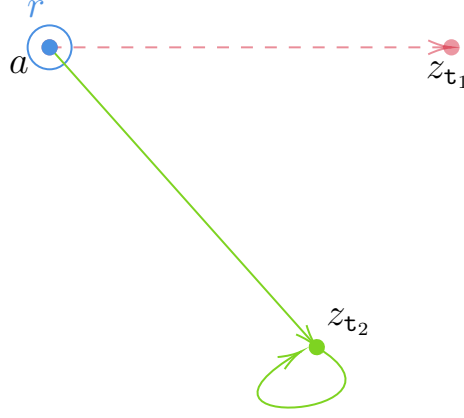
We will use the abbreviation **V**- and **F**-chase for vacuum and frugal chase and **V**- and **F**-derivation for vacuum and frugal derivations respectively. Continuing on the Example 13, we demonstrate how the frugal and the vacuum chase will behave in this case:

Example “FRUGALPHA” 17 (continued from Example 13): Here is a **F**-derivation \mathcal{D}' from (F, \mathcal{R}) :

\emptyset	F	$Z'_0 = F$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto a\})$	$F'_1 = F_1 = F \cup \{p(a, z_{\mathfrak{t}_1})\}$	$Z'_1 = F'_1$	1
$\mathfrak{t}_2 = (R_2, \{x \mapsto a, y \mapsto z_{\mathfrak{t}_1}\})$	$F'_2 = F'_1 \cup \{p(a, z_{\mathfrak{t}_2}), p(z_{\mathfrak{t}_2}, z_{\mathfrak{t}_2})\}$	$Z'_2 = F'_2 \setminus \{p(a, z_{\mathfrak{t}_1})\}$	2

We provide a representation of $Z^{\mathcal{D}'}$:

⁸The complexity of the graph isomorphism problem is in NP, and it has not been shown NP-complete, whereas retraction is NP-complete.



We can see that \mathcal{D}' is terminating since there is no further trigger from \mathcal{R} that is \mathbf{R} -applicable on Z_2 . Lastly, notice that \mathcal{D}' is also a (terminating) \mathbf{V} -derivation, since in this case $\mathbf{V}(\mathbf{t}_1, Z_0) = \mathbf{F}(\mathbf{t}_1, Z_0)$ and $\mathbf{V}(\mathbf{t}_2, Z_1) = \mathbf{F}(\mathbf{t}_2, Z_1)$. ■

From the relation $\mathbf{V}(\mathbf{t}, Z) \subseteq \mathbf{F}(\mathbf{t}, Z)$ (for every \mathbf{t} and Z), we conclude that every terminating \mathbf{F} -derivation (such as the one in the above example) corresponds to a terminating \mathbf{V} -derivation from the same knowledge base (by applying triggers with the same order we produce smaller active factbases). This observation will be useful when classifying chase variants with respect to termination in the next section. In our examples we mainly use derivations which belong to both the frugal & the vacuum chase, because it facilitates the deduction of several results. Notice however that there is no containment relation between those two very similar chase variants, i.e. a \mathbf{V} -derivation is not necessarily a \mathbf{F} -derivation and vice versa.

4.3.2 Parallel Chase & Core Chase

Except from the capability of deleting redundant atoms from the active factbase, the formalism introduced in Definition 4.1 can also represent an important property of potential chase variants, namely the parallel application of rules. This is particularly useful when combined with a breadth-first prioritization of rule applications. Therefore we provide the following definition of a synchronous (breadth-first) derivation, based on \mathbf{R} -applicability, which can be seen as a superclass of derivations that can serve as a general platform for the definition of

different chase algorithms. The choice of **R**-applicability condition is in accordance with how the *parallel* and *core* chase were first introduced in the seminal paper [18]. We remind that we call *rank mark* the last element of each rank in a rank compatible derivation.

Definition 4.13 (Synchronous Derivation). Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ be a derivation. We call \mathcal{D} a *synchronous derivation* if for every $i > 0$,

- $\text{rank}(\mathfrak{t}_i) \geq \text{rank}(\mathfrak{t}_{i-1})$, i.e. \mathcal{D} is rank compatible,
- the trigger \mathfrak{t}_i is **R**-applicable on Z_{i-1} and
- if there exist at least two different triggers that are **R**-applicable on Z_{i-1} , it holds that $Z_i = Z_{i-1}$.

Then, for every two consecutive rank marks D_j and D_i in \mathcal{D} , we denote the *rank's union of transitory factbases* with $\widehat{F}_i = \bigcup_{j < \ell \leq i} F_\ell$. ⊢

In other words, in a synchronous derivation the active factbase can only change in the rank marks and is static otherwise. Note that an equal way of specifying \widehat{F}_i is as $Z_j \cup \text{op}(\mathfrak{t}_{j+1}) \cup \dots \cup \text{op}(\mathfrak{t}_i)$. As a result of the conditions stated above, we cannot change the active factbase unless we have already applied all **R**-applicable triggers. This conforms with the concept of a breadth-first derivation, as we have seen with the α -operator in Section 2.2 (albeit based on applicability rather than **R**-applicability), with the **E**-chase in the previous section (using **E**-applicability) and with the general Definition 4.8. Yet in this case, the concept is enforced by a condition imposed on the active factbase.

Because there is no constraint concerning the treatment of the active factbases in rank marks, the whole class of synchronous derivations constitutes a chase variant with a rather vague functionality. Nevertheless, it is useful for classification purposes as all the chase variants that are included in the class of synchronous derivations have several common properties. It is easy to show the following:

Proposition 4.2. Let X be a chase variant contained in the class of synchronous derivations. Then X is a breadth-first chase variant, i.e. every X -derivation is a breadth-first X -derivation. ♣

Proof: We use contradiction. Let \mathcal{D} be an X -derivation that is not a breadth-first X -derivation. Then there is a rank k , and a prefix \mathcal{D}^k of \mathcal{D} which includes all elements of rank at most k in \mathcal{D} such that there is a trigger \mathfrak{t} of rank k in \mathcal{D} that is X -applicable on \mathcal{D}^k . So \mathfrak{t} is \mathbf{R} -applicable on $Z^{\mathcal{D}^k}$. But since \mathfrak{t} does not appear in \mathcal{D} , it is not \mathbf{R} -applicable on $Z^{\mathcal{D}^{k-1}}$. So there is a retraction h from $\text{op}(\mathfrak{t}) \cup Z^{\mathcal{D}^{k-1}}$ to $Z^{\mathcal{D}^{k-1}}$. Let i be the length of \mathcal{D}^k and let g be the retraction from \widehat{F}_i to $Z^{\mathcal{D}^k}$. We know that $Z^{\mathcal{D}^{k-1}} \subseteq \widehat{F}_i$. So we have $g \circ h(\text{op}(\mathfrak{t})) \subseteq Z^{\mathcal{D}^k}$. Moreover, since $\text{sp}(\mathfrak{t}) \in Z^{\mathcal{D}^k}$ we also know that $\text{dom}(g) \cap \text{var}(\text{sp}(\mathfrak{t})) = \emptyset$. Therefore also $\text{dom}(g \circ h) \cap \text{var}(\text{sp}(\mathfrak{t})) = \emptyset$, hence $g \circ h$ is a retraction from $\text{op}(\mathfrak{t}) \cup Z^{\mathcal{D}^k}$ to $Z^{\mathcal{D}^k}$. This means that \mathfrak{t} is not \mathbf{R} -applicable on $Z^{\mathcal{D}^k}$, which is a contradiction. □

Now we use the the synchronous derivation platform to present two more non-monotonic chase variants (originally introduced in [18]). When the specification of the active factbases in the rank marks involves simply adding the atoms produced at this breadth-first level, the resulting chase is called *parallel chase*.

Definition 4.14 (Parallel Chase). A **parallel derivation** is any synchronous derivation $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ from (F, \mathcal{R}) where for every rank mark k we have $Z_k = \widehat{F}_k$. ⊥

Again, we will mostly be using the abbreviation **P**-derivation and **P**-chase when discussing this chase variant.

Example 18: We have that $F = \{p(a, b)\}$ and $\mathcal{R} = \{R\}$ where:

$$R = p(x, y) \rightarrow \exists z \, p(y, z) \wedge p(z, z) \wedge p(z, x)$$

Below is the **P**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	$F_0 = F$	$Z_0 = F$	0
$\mathbf{t}_1 = (R, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{p(b, z_{\mathbf{t}_1}), p(z_{\mathbf{t}_1}, z_{\mathbf{t}_1}), p(z_{\mathbf{t}_1}, a)\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R, \{x \mapsto b, y \mapsto z_{\mathbf{t}_1}\})$	$F_2 = Z_1 \cup \{p(z_{\mathbf{t}_1}, z_{\mathbf{t}_2}), p(z_{\mathbf{t}_2}, z_{\mathbf{t}_2}), p(z_{\mathbf{t}_2}, b)\}$	$Z_2 = Z_1$	
$\mathbf{t}_3 = (R, \{x \mapsto z_{\mathbf{t}_1}, y \mapsto a\})$	$F_3 = Z_1 \cup \{p(a, z_{\mathbf{t}_3}), p(z_{\mathbf{t}_3}, z_{\mathbf{t}_3}), p(z_{\mathbf{t}_3}, z_{\mathbf{t}_1})\}$	$Z_3 = F_2 \cup F_3$	2
$\mathbf{t}_4 = (R, \{x \mapsto z_{\mathbf{t}_1}, y \mapsto z_{\mathbf{t}_2}\})$	$F_4 = Z_3 \cup \{p(z_{\mathbf{t}_2}, z_{\mathbf{t}_4}), p(z_{\mathbf{t}_4}, z_{\mathbf{t}_4}), p(z_{\mathbf{t}_4}, z_{\mathbf{t}_1})\}$	$Z_4 = Z_3$	
$\mathbf{t}_5 = (R, \{x \mapsto z_{\mathbf{t}_3}, y \mapsto z_{\mathbf{t}_1}\})$	$F_5 = Z_3 \cup \{p(z_{\mathbf{t}_1}, z_{\mathbf{t}_5}), p(z_{\mathbf{t}_5}, z_{\mathbf{t}_5}), p(z_{\mathbf{t}_5}, z_{\mathbf{t}_3})\}$	$Z_5 = F_4 \cup F_5$	3

In Figure 4.5 we see a representation of $Z^{\mathcal{D}}$. \mathcal{D} is a terminating **P**-derivation of

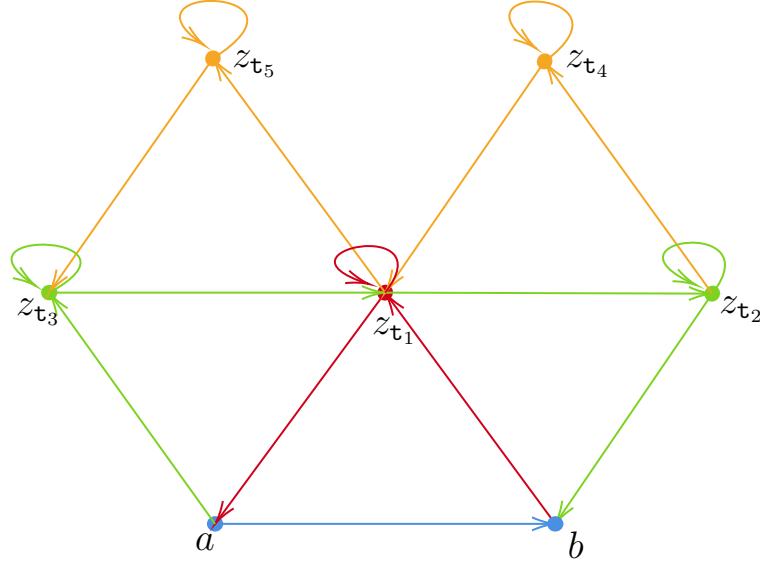


Figure 4.5: Example 18, representation of $Z^{\mathcal{D}}$.

depth 3, as there is no trigger from \mathcal{R} that is **R**-applicable on Z_5 . ■

The parallel chase in itself can be regarded as a deterministic breadth-first version of the restricted chase, however its capability of detecting redundancies is even weaker, as **R**-derivations have the advantage that applicability is tested against factbases that include whatever new information was added in the current rank. On the other hand, based on the concept of a synchronous derivation, we can define the chase variant which is known to filter out the most redundancy:

Definition 4.15 (Core Chase). A **core derivation** is any synchronous derivation $\mathcal{D} = (\mathbf{t}_*, F_*, Z_*)$ from (F, \mathcal{R}) where for every rank mark k we have $Z_k = \text{core}(\hat{F}_k)$. ⊥

We abbreviate with C-derivation and C-chase. To demonstrate how the core chase works, we resume discussion around Example 3.

Example “GENERIC” 19 (continued from Examples 3, 4 and 8): We will be working on the same knowledge base, while specifying a different derivation this time. We remind that $F = \{p(a, b), p(c, d), r(e)\}$ and \mathcal{R} is the following set of rules:

$$R_1 = p(x, y) \wedge r(z) \rightarrow p(y, z)$$

$$R_2 = p(x, y) \wedge p(y, z) \rightarrow \exists u p(z, u)$$

$$R_3 = p(x, y) \wedge p(x, z) \rightarrow p(y, z)$$

We display a C-derivation \mathcal{D}' from (F, \mathcal{R}) . Notice that we use the same triggers in \mathcal{D}' as were used in \mathcal{D} . But the order of the triggers is different in this derivation. This results to different factbases, which are denoted with a prime (e.g. F'_i, Z'_i), to differentiate with those found in \mathcal{D} .

\emptyset	F	$Z'_0 = F$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b, z \mapsto e\})$	$F'_1 = F \cup \{p(b, e)\}$	$Z'_1 = F$	
$\mathbf{t}_4 = (R_1, \{x \mapsto c, y \mapsto d, z \mapsto e\})$	$F'_2 = F'_1 \cup \{p(d, e)\}$	$Z'_2 = F$	
$\mathbf{t}_5 = (R_3, \{x \mapsto a, y \mapsto b, z \mapsto b\})$	$F'_3 = F'_2 \cup \{p(b, b)\}$	$Z'_3 = F$	
$\mathbf{t}_6 = (R_3, \{x \mapsto c, y \mapsto d, z \mapsto d\})$	$F'_4 = F'_3 \cup \{p(d, d)\}$	$Z'_4 = F'_4$	1
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto b, z \mapsto e\})$	$F'_5 = Z'_4 \cup \{p(e, u_{\mathbf{t}_2})\}$	$Z'_5 = Z'_4$	
$\mathbf{t}_7 = (R_2, \{x \mapsto c, y \mapsto d, z \mapsto e\})$	$F'_6 = F'_5 \cup \{p(e, u_{\mathbf{t}_7})\}$	$Z'_6 = Z'_4$	
$\mathbf{t}_8 = (R_1, \{x \mapsto b, y \mapsto e, z \mapsto e\})$	$F'_7 = F'_6 \cup \{p(e, e)\}$	$Z'_7 = Z'_4 \cup \{p(e, e)\}$	2

Here Z'_7 is a core of $F'_5 \cup F'_6 \cup F'_7$. In Figure 4.6 we can see $F^{\mathcal{D}'}$, where atoms are colored with respect to rank (blue, red, green) and dotted lines are used

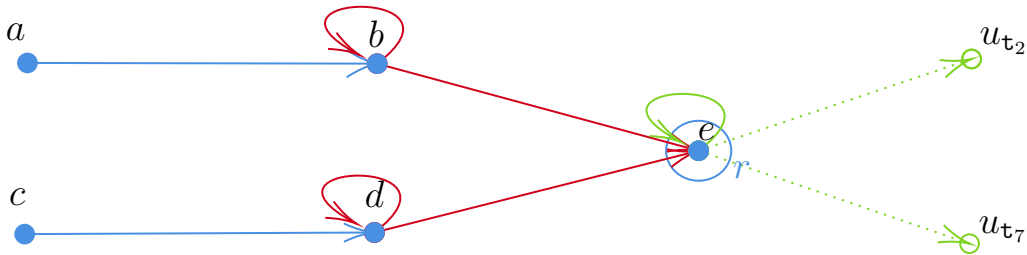


Figure 4.6: Example “GENERIC”, C-chase: representation of $F^{\mathcal{D}'}$.

to represent the two atoms that are produced but not added to the final active factbase (as they do not belong to the core). ■

The C-derivation of the above example is terminating, since there are no more possible rule applications on Z'_7 . Notice that in the core chase, we apply a trigger if it is **R**-applicable to the active factbase. When Z is a core and $\text{op}(\tau)$ an atomset, then there exists a homomorphism from $Z \cup \text{op}(\tau)$ to Z if and only if there exists a retraction from $Z \cup \text{op}(\tau)$ to Z . This is why **R**-applicability is sufficient.

4.3.3 Submonotonicity

So far we have discussed monotonic and non-monotonic chase variants. There is however one more characterization, which divides the chase landscape in another fairly interesting and, as it turns out, important manner. A non-monotonic derivation can nonetheless have the property that the active factbase is increasing monotonically. As we will see in Subsection 5.2.3, several key notions are related to this property.

Definition 4.16. A derivation $\mathcal{D} = (\tau_*, F_*, Z_*)$ is *submonotonic* if for every i holds that $Z_i \subseteq Z_{i+1}$. A chase variant **X** is *submonotonic* if every **X**-derivation is submonotonic. \dashv

It is obvious that every monotonic derivation is a submonotonic derivation. Therefore we can say that the **O**-, **SO**-, **R**- and **E**-chase are submonotonic chase variants. From the three chase variants that we introduced in this section, only the **P**-chase is submonotonic. Indeed from Example 13.17 we can see that the **F**-chase and the **V**-chase are not submonotonic. Furthermore, in the **P**-chase by definition we never remove atoms from the active factbase. On the other hand here is an example showing that the **C**-chase is not submonotonic:

Example “EQUICORE” 20 (continued from Example 11): We remind that $F = \{p(a, b)\}$ and \mathcal{R} is the following set of rules:

$$R_1 = p(x, y) \rightarrow \exists z p(y, z)$$

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

Below is the C-derivation \mathcal{D}' from (F, \mathcal{R}) :

\emptyset	$F'_0 = F$	$Z'_0 = F'_0$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F'_1 = F \cup \{p(b, z_{\mathfrak{t}_1})\}$	$Z'_1 = F'_1$	1
$\mathfrak{t}_2 = (R_2, \{x \mapsto a, y \mapsto b, z \mapsto z_{\mathfrak{t}_1}\})$	$F'_2 = F'_1 \cup \{p(b, b)\}$	$Z'_2 = F'_2 \setminus \{p(b, z_{\mathfrak{t}_1})\}$	2

We see that the derivation \mathcal{D}' is very similar with the E-derivation \mathcal{D} from (F, \mathcal{R}) which is presented in the previous section. Indeed it holds that $\text{trig}(\mathcal{D}') = \text{trig}(\mathcal{D})$, $F'_i = F_i$ for every i , $Z'_0 = Z_0$ and $Z'_1 = Z_1$. The only difference is that there is one element removed from the final factbase Z'_2 . We have that \mathcal{D}' is terminating as a C-derivation, since there is no trigger from \mathcal{R} that is R-applicable on Z'_2 . Lastly, \mathcal{D}' is not submonotonic, as the atom $p(b, z_{\mathfrak{t}_1})$ belongs to Z'_1 but not to Z'_2 . ■

4.4 Comparing Chase Variants

Having specified a general platform in which a multitude of chase variants can be defined, the need to find appropriate criteria of comparison of chase variants emerges. In the first part of this section we define two such criteria, namely *termination* and *elimination of redundancy*. In the second part we explore whether the breadth-first approach to forward chaining is sufficient when researching properties related to depth and termination of derivations. We find that by restricting a chase variant to only breadth-first derivations we obtain a different behavior towards termination. We conclude that this restriction fundamentally changes the nature of the chase variant and therefore for every chase variant X, the chase variant comprised by all breadth-first X-derivations (named bf-X) merits independent investigation. Figure 4.10, which can be found at the end of this section, summarizes our findings with regards to termination.

4.4.1 Termination & Elimination of Redundancy

The first intuitive measure of comparison between chase variants relates to terminating derivations. Notice that, as showcased by the Example 12, the fact

that one **R**-derivation from a given knowledge base terminates, does not imply that every exhaustive **R**-derivation from the same knowledge is finite (i.e. terminating). This observation motivates the first important division between chase variants:

Definition 4.17. A chase variant X is *termination-order independent* if for every knowledge base (F, \mathcal{R}) , if there exists a terminating X -derivation \mathcal{D} from (F, \mathcal{R}) , then every exhaustive X -derivation from (F, \mathcal{R}) is terminating. A chase variant that is not termination-order independent is called *termination-order dependent*. \dashv

The **O**-chase is termination-order independent because by definition the choice of which trigger to apply cannot cancel the applicability of other triggers. In the Appendix (Section (B)) we show that **SO**-chase is termination-order independent. The **E**-chase and the **C**-chase terminate if and only if there is a finite universal model of the knowledge base, so they are termination-order independent, as is the **P**-chase where rules can be considered to be applied synchronously. We have already seen a counter-example⁹ that shows that the **R**-chase is termination-order dependent. And in the next subsection we will provide a counter-example¹⁰ that shows that the **F**-chase and the **V**-chase are termination-order dependent.

In chase variants that are termination-order dependent, the quest for optimal ways to find a terminating derivation from a knowledge base (if there is one), has interest and value. But in our study we are focusing in properties related to when *all* derivations terminate. So continuing this discussion, the following definition provides a measure of comparison of chase variants with regard to termination:

Definition 4.18 (Strength of Termination). Let X and Y be two chase variants. We say that X is *as strong as* Y with respect to *termination*, if for every knowledge base (F, \mathcal{R}) , if every exhaustive Y -derivation from (F, \mathcal{R}) is terminating, then also every exhaustive X -derivation from (F, \mathcal{R}) is terminating. If X is as

⁹Example 12.

¹⁰Example 24.

strong as Y but Y is not as strong as X wrt termination, we will say that X is *stronger than* Y wrt *termination*. If X is as strong as Y and Y is as strong as X , we will say that X and Y are *equivalent* wrt *termination*. \dashv

We will use the notation $X \geq_t Y$, $X >_t Y$ and $X \equiv_t Y$ to say that X is respectively as strong as Y , stronger than Y and equivalent with Y wrt termination. It is evident that strength of termination is a transitive relation. The following proposition delineates the well known classification of the monotonic chase variants with respect to termination. At the end of this section we will provide a table with the respective partial order of all the concerned chase variants with respect to termination.

Proposition 4.3 (Monotonic Chase Termination Classification [37, 55]). It holds that $\mathbf{E}\text{-chase} >_t \mathbf{R}\text{-chase} >_t \mathbf{SO}\text{-chase} >_t \mathbf{O}\text{-chase}$. \clubsuit

Termination is an important aspect of the chase, but it is an incomplete criterion of comparison of chase variants. That is because in practice, the computational complexity is related to the length of derivations and the potential expansion of the (active) factbase. Those two parameters can fluctuate greatly independently of whether a derivation terminates. Moreover, in a practical setting, we do not know which derivations terminate, hence we do not only work with terminating derivations.

All the chase variants presented in this thesis aim towards constructing a universal model of the knowledge base. Although strength in termination as defined above does certainly give a measure of the efficiency of chase variants in constructing universal models, they do not provide information concerning the length of derivations or the size of resulting factbases. As a more precise criterion of (computational) comparison of chase variants, we specify the following:

Definition 4.19 (Elimination of Redundancy). Let X and Y be two chase variants. X is *as strong as* Y *in eliminating redundancy* if for every finite X -derivation \mathcal{D} from a knowledge base (F, \mathcal{R}) , there is a Y -derivation \mathcal{D}' from (F, \mathcal{R}) such that

- i. $\text{trig}(\mathcal{D})$ is a subset (not necessarily a subsequence) of $\text{trig}(\mathcal{D}')$,
- ii. \mathcal{D}' is minimal with the above property, i.e. there is no Y-derivation \mathcal{D}'' with $\text{trig}(\mathcal{D}) \subseteq \text{trig}(\mathcal{D}'') \subset \text{trig}(\mathcal{D}')$ and
- iii. $Z^{\mathcal{D}} \subseteq Z^{\mathcal{D}'}$.

If X is as strong as Y but Y is not as strong as X in eliminating redundancy, we will say that X is *stronger than* Y (in eliminating redundancy). Lastly if X is as strong as Y and Y is as strong as X in eliminating redundancy, we will say that X and Y are *equivalent* in eliminating redundancy. \dashv

We will use the notation $X \geq_r Y$, $X >_r Y$ and $X \equiv_r Y$ to say that X is respectively as strong as Y, stronger than Y and equivalent with Y in eliminating redundancy. So if X is as strong as Y, we know that for every X-derivation there is a Y-derivation which includes the same rule applications without eliminating more redundant atoms from the active factbase. But what is the connection between strength in elimination of redundancy and strength in termination?

Proposition 4.4. Let X, Y be chase variants such that X is stronger than Y wrt termination. Then Y is not as strong as X in eliminating redundancy. Symbolically, $X >_t Y$ implies $X \not\leq_r Y$.

Proof: We know that there exists a knowledge base (F, \mathcal{R}) such that all X-derivations from (F, \mathcal{R}) are terminating but there is a Y-derivation \mathcal{D} from (F, \mathcal{R}) that is not terminating. Let n be the maximum length of any X-derivation from (F, \mathcal{R}) . Then the prefix of \mathcal{D} of length $n + 1$, according to the above definition, provides a counter-example that shows that X is not as strong as Y in eliminating redundancy. \square

Notice that two chase variants can be incomparable with respect to the elimination of redundancy. In Example 13.17 we specify the F-derivation \mathcal{D}' . It is easy to see that the E-derivation with the same sequence of associated triggers necessarily results in a bigger active factbase. Hence the E-chase is not as strong as the F-chase in eliminating redundancy. Similarly, we know that E-chase is stronger than the F-chase wrt termination. Hence from the above proposition the F-chase is not as strong as the E-chase in eliminating redundancy.

Most definitions of chase variants allow for very different derivations from the same knowledge base. Assuming our goal is to find terminating derivations, we are looking for optimal strategies towards that goal. These strategies are directed towards narrowing down the class of derivations that we are considering, hence reducing the chase variant under examination. An intuitive and popular strategy in such cases is the breadth-first forward chaining.

4.4.2 The Breadth-first Approach

In this subsection we investigate how restricting our interest to breadth-first derivations affects termination. Informally speaking, a derivation which is not breadth-first, can be seen as a derivation where the rules are applied with priority on a subset of the knowledge base. This can have a significant effect, in several different occasions. First note that in Datalog all exhaustive O-derivations have the same length but not necessarily the same depth, as illustrated by the following example.

Example “DATALOG” 21: Let $F = \{p(a)\}$ and $\mathcal{R} = \{R_1, R_2, R_3\}$ where $R_1 = p(x) \rightarrow q(x)$, $R_2 = q(x) \rightarrow r(x)$, $R_3 = p(x) \rightarrow r(x)$. Here is the O-derivation \mathcal{D}_1 :

\emptyset	F	$Z_0 = F$
$\mathfrak{t}_1 = (R_1, \{x \mapsto a\})$	$F_1 = F \cup \{q(a)\}$	F_1
$\mathfrak{t}_2 = (R_2, \{x \mapsto a\})$	$F_2 = F_1 \cup \{r(a)\}$	F_2
$\mathfrak{t}_3 = (R_3, \{x \mapsto a\})$	F_2	F_2

and here is the O-derivation \mathcal{D}_2 :

\emptyset	F	$Z_0 = F$	0
\mathfrak{t}_1	F_1	F_1	
\mathfrak{t}_3	F_2	F_2	1
\mathfrak{t}_2	F_2	F_2	2

We can see that both derivations are exhaustive, however the depth of \mathcal{D}_1 is 2 whereas the depth of \mathcal{D}_2 is 1. ■

Nevertheless, among all exhaustive X-chase¹¹ derivations in a Datalog setting (i.e. Datalog ruleset and variable-free factbase), the class of breadth-first derivations are of minimal depth. We will show that this remains true for the oblivious and semi-oblivious chase derivations when we expand the setting to existential rules (i.e. considering any knowledge base of positive existential rules). But first we provide a definition and discuss several useful properties of the SO-chase and the O-chase.

Definition 4.20. Let $t_1 = (R, \pi)$, $t_2 = (R, \pi')$ be two triggers. We say that t_1 and t_2 are *SO-equivalent* if π and π' agree in their mappings of the frontier variables of R . \dashv

Hence when constructing a SO-derivation, we are allowed to choose at most one trigger from the corresponding SO-equivalence class. In particular:

Remark 4.3. The following observations result directly from the above definition:

- i. Let t and t' be two SO-equivalent triggers. Then there is an isomorphism $\tau : \text{nul}(\text{op}(t)) \rightarrow \text{nul}(\text{op}(t'))$ from $\text{op}(t)$ to $\text{op}(t')$.
- ii. Let \mathcal{D} be an O-derivation. \mathcal{D} is a SO-derivation if and only if for every pair of triggers $t_1 \neq t_2$ where $t_1, t_2 \in \text{trig}(\mathcal{D})$ holds that t_1 is not SO-equivalent with t_2 .
- iii. Let t be a trigger that is O-applicable on a derivation \mathcal{D} . It holds that t is also SO-applicable on \mathcal{D} if and only if there does not exist a trigger $t' \in \text{trig}(\mathcal{D})$ such that t' is SO-equivalent with t . \clubsuit

Now we consider the “reshuffling” of a derivation in a rank compatible fashion. We want to apply this in particular to O- and SO-derivations. Nonetheless we prove a proposition that concerns any derivation. Notice that every derivation has a rank compatible prefix, since in the worst case rank compatibility holds for the first two elements. We show that given a finite derivation, any trigger of a rank that corresponds to its rank compatible prefix is O-applicable to this

¹¹here $X \in \{\text{O}, \text{SO}, \text{R}, \text{E}, \text{P}, \text{F}, \text{C}\}$.

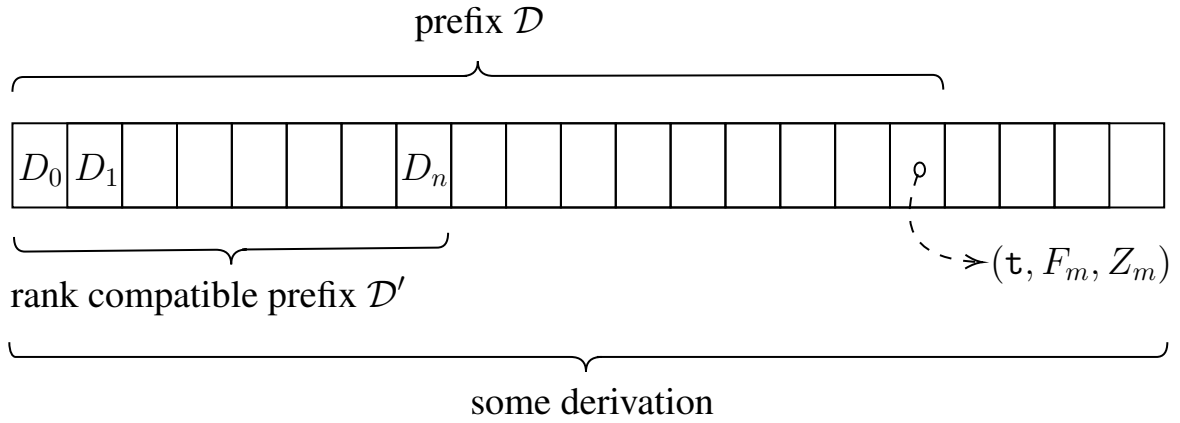


Figure 4.7: Representation of the *rank compatible trigger sorting*.

prefix, even if it appears much later in the derivation. Furthermore the atoms produced by this trigger when applied to the prefix will have lower or equal ranks in comparison with their ranks in the original derivation. Figure 4.7 provides an intuition of this operation, using the notation of the following proposition. In the general case of an X-chase, we do not know if the trigger is X-applicable or how its application will affect the active factbase. But for O- and SO-derivations, we can use this operation repeatedly until we arrive at a rank compatible derivation.

Proposition 4.5 (Rank compatible trigger sorting). Let \mathcal{D} be a finite derivation with \mathfrak{t} its last trigger and $\mathcal{D}' = D_0, D_1, \dots, D_n$ a rank compatible prefix of \mathcal{D} such that

- $$\begin{aligned} \text{i)} \quad & \max \{ \text{rank}(\mathbf{t}') \mid \mathbf{t}' \in \mathbf{trig}(\mathcal{D}') \} \leq \text{rank}(\mathbf{t}) \text{ and} \\ \text{ii)} \quad & \min \{ \text{rank}(\mathbf{t}') \mid \mathbf{t}' \in \mathbf{trig}(\mathcal{D}) \setminus \mathbf{trig}(\mathcal{D}') \} \geq \text{rank}(\mathbf{t}). \end{aligned}$$

Then \mathfrak{t} is \mathbf{O} -applicable on \mathcal{D}' and for any derivation $\mathcal{D}'' = D_0, D_1, \dots, D_n, (\mathfrak{t}, F''_{n+1}, Z''_{n+1})$ holds that $rank_{\mathcal{D}''}(A) \leq rank_{\mathcal{D}}(A)$ for every $A \in \mathbf{op}(\mathfrak{t})$.

Proof: Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$. We know that \mathfrak{t} is applicable on $F^{\mathcal{D}'}$ because all atoms of smaller rank than \mathfrak{t} are already produced in \mathcal{D}' , so $\text{sp}(\mathfrak{t}) \subseteq F^{\mathcal{D}'}$. And also $\mathfrak{t} \notin \text{trig}(\mathcal{D}')$, so \mathfrak{t} is O-applicable on \mathcal{D}' . In addition we have that $\text{rank}_{\mathcal{D}}(\mathfrak{t}) = \text{rank}_{\mathcal{D}''}(\mathfrak{t})$ since the rank of a trigger depends only on the ranks of its supporting atoms (which does not change here). Then we have three cases:

- A is produced by \mathfrak{t} in \mathcal{D} . Then A has the same rank in \mathcal{D}'' , which is the rank of \mathfrak{t} .
- A is produced by a trigger \mathfrak{t}_i where $i \leq n$ in \mathcal{D} . Then A is already produced by the time \mathfrak{t} is applied in both \mathcal{D} and \mathcal{D}'' , and it is produced in their common prefix, so $\text{rank}_{\mathcal{D}''}(A) = \text{rank}_{\mathcal{D}}(A)$.
- A is produced by a trigger \mathfrak{t}_i where $i \geq n + 1$ in \mathcal{D} . Then by ii) we know that $\text{rank}_{\mathcal{D}}(\mathfrak{t}) \leq \text{rank}_{\mathcal{D}}(\mathfrak{t}_i)$, so also $\text{rank}_{\mathcal{D}''}(\mathfrak{t}) \leq \text{rank}_{\mathcal{D}}(\mathfrak{t}_i)$, so $\text{rank}_{\mathcal{D}''}(A) \leq \text{rank}_{\mathcal{D}}(A)$.

Hence $\text{rank}_{\mathcal{D}''}(A) \leq \text{rank}_{\mathcal{D}}(A)$ holds in all cases. \square

In accordance with this last result, given an **O**-derivation that is not rank compatible, if we try to rearrange it in a rank compatible manner, by moving all triggers to their respective rank, we will obtain an **O**-derivation that is of smaller or equal depth. Following point ii. of Remark 4.3, we can use the same technique to rearrange **SO**-derivations. In addition, it is important to recognize that this process optimizes the depth of derivations. Note that for other chase variants, rearranging triggers is not as trivial, as there are restrictions in the applicability of triggers and in the evolution of the active base. Now by employing these findings, we will show that the breadth-first approach is useful when considering problems related to termination in the oblivious and the semi-oblivious chase.

Proposition 4.6. For each terminating **O**-derivation (respectively **SO**-derivation) from (F, \mathcal{R}) there exists a breadth-first terminating **O**-derivation (respectively **SO**-derivation) from (F, \mathcal{R}) of smaller or equal depth.

Proof: **Case O:** Let \mathcal{D} be a terminating **O**-derivation from (F, \mathcal{R}) . Let \mathcal{T} be a re-ordering of $\text{trig}(\mathcal{D})$ according to rank (i.e. rank compatible). By definition of rank, rearranging the sequence of triggers according to rank compatibility does not affect **O**-applicability. Therefore, let \mathcal{D}' be an **O**-derivation from

(F, \mathcal{R}) such that $\text{trig}(\mathcal{D}') = \mathcal{T}$. If \mathcal{D}' is not terminating, then there is a new trigger \mathfrak{t} **O**-applicable on $Z^{\mathcal{D}'}$. But $Z^{\mathcal{D}'} = F^{\mathcal{D}'} = F^{\mathcal{D}} = Z^{\mathcal{D}}$ so \mathfrak{t} is **O**-applicable on \mathcal{D} as well. Then \mathcal{D} is not terminating, but that is a contradiction. Therefore \mathcal{D}' is terminating. If \mathcal{D}' is not breadth-first, then there is a new trigger \mathfrak{t} **O**-applicable in some intermediate rank on a subset Z' of $Z^{\mathcal{D}'}$. But then \mathfrak{t} is also **O**-applicable on $Z^{\mathcal{D}'}$ and following the same argumentation as above we obtain again that \mathcal{D} is not terminating, which is a contradiction. Therefore \mathcal{D}' is breadth-first. And since \mathcal{D}' was obtained based on rank compatible trigger sorting, from Proposition 4.5, we obtain that it is of smaller or equal depth than \mathcal{D} .

Case SO: Let \mathcal{D} be a terminating **SO**-derivation from (F, \mathcal{R}) . As per Remark 4.3.iii, when **O**-applicability is secured, the condition for **SO**-applicability is non-**SO**-equivalence. Hence, we can rearrange $\text{trig}(\mathcal{D})$ in a rank compatible manner, obtaining \mathcal{T} , and then we have a **SO**-derivation \mathcal{D}' from (F, \mathcal{R}) with $\text{trig}(\mathcal{D}') = \mathcal{T}$. So \mathcal{D}' is rank compatible and moreover it is terminating (otherwise \mathcal{D} would also not be terminating). And from Proposition 4.5, we obtain that \mathcal{D}' is of smaller or equal depth than \mathcal{D} . However we do not know if \mathcal{D}' is breadth-first. Nevertheless we can transform it to a breadth-first derivation. This is achieved by starting from the lowest ranks and verifying if all **SO**-applicable triggers are indeed applied. If not, we add the trigger one by one to the derivation. Each time that we add a trigger, we remove a **SO**-equivalent trigger of a higher rank. Since they do not have exactly the same output (albeit isomorphic), the following triggers need to change, following the renaming of the new variable. This process can only decrease the ranks of triggers and atoms and hence also the overall depth. A detailed proof is given in the Appendix(Section (B)). \square

In the previous subsection we discussed termination-order independence and saw that **O**-chase and **SO**-chase have this property. On the other hand, as demonstrated in Example 12 at the end of Section 4.2, in the restricted chase the order of application of the rules can have a decisive impact on whether a derivation will terminate or not. From that example we saw that even chang-

ing the order of the rule applications within a certain rank, the property of (non-)termination is influenced. One can have an intuitive tendency to consider the breadth-first approach as the most efficient in forward chaining. This is a result of the premise that by ignoring a part of the knowledge base, we lose information that would have been beneficial when performing rule applications of higher ranks. As we will see, this is not true in general. In some chase variants there can be atoms which when produced early on in a derivation, they hinder the detection of redundancies later.

As a corollary of Proposition 4.6 combined with the fact that both the **O**-chase as well as the **SO**-chase are termination-order independent, we conclude that it suffices to consider breadth-first derivations when investigating problems related to the termination and depth of derivations for oblivious and semi-oblivious chase. In addition, the equivalent chase is by definition breadth-first, as is every chase variant which is based on the concept of a synchronous derivation (from Proposition 4.2). This leaves the restricted, the frugal and the vacuum chase as the chase variants where we have to question the effectiveness of the breadth-first approach.

Example 22: Let $F = \{p(a, b)\}$ and \mathcal{R} the following set of rules:

$$R_1 = p(x, y) \rightarrow \exists z p(y, z)$$

$$R_2 = p(x, y) \rightarrow \exists z q(y, z)$$

$$R_3 = q(y, z) \rightarrow p(y, y)$$

Here is a terminating **R**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathbf{t}_1 = (R_2, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{q(b, z_{\mathbf{t}_1})\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R_3, \{x \mapsto b, y \mapsto z_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{p(b, b)\}$	$Z_2 = F_2$	2

The trigger $\mathbf{t} = (R_1, \{x \mapsto a, y \mapsto b\})$ is **R**-applicable on Z_1 , and it is of rank 1, whereas \mathbf{t}_2 is of rank 2. Therefore, \mathcal{D} is not breadth-first. If we would have made that rule application before applying \mathbf{t}_2 and then completed all rule applications up to rank 2 (in order to make it breadth-first) we would obtain \mathcal{D}' :

\emptyset	F	$Z'_0 = F$	0
$\mathbf{t}_1 = (R_2, \{x \mapsto a, y \mapsto b\})$	$F'_1 = F \cup \{q(b, z_{\mathbf{t}_1})\}$	$Z'_1 = F'_1$	
$\mathbf{t} = (R_1, \{x \mapsto a, y \mapsto b\})$	$F'_2 = F'_1 \cup \{p(b, z_{\mathbf{t}})\}$	$Z'_2 = F'_2$	1
$\mathbf{t}_2 = (R_3, \{x \mapsto b, y \mapsto z_{\mathbf{t}_1}\})$	$F'_3 = F'_2 \cup \{p(b, b)\}$	$Z'_3 = F'_3$	
$\mathbf{t}' = (R_1, \{x \mapsto b, y \mapsto z_{\mathbf{t}}\})$	$F'_4 = F'_3 \cup \{p(z_{\mathbf{t}}, z_{\mathbf{t}'})\}$	$Z'_4 = F'_4$	
$\mathbf{t}'' = (R_2, \{x \mapsto b, y \mapsto z_{\mathbf{t}}\})$	$F'_5 = F'_4 \cup \{q(z_{\mathbf{t}}, z_{\mathbf{t}''})\}$	$Z'_5 = F'_5$	2

It is easy to see that \mathcal{D}' is not a terminating **R**-derivation as $\mathbf{t}''' = (R_1, \{x \mapsto z_{\mathbf{t}}, y \mapsto z_{\mathbf{t}'}\})$ is **R**-applicable on $Z^{\mathcal{D}'}$. And furthermore, if we do another round of similar rule applications respecting the breadth-first conditions, we will be found with a similar non-terminating **R**-derivation. In other words, while there exists a terminating **R**-derivation from (F, \mathcal{R}) , every breadth-first **R**-derivation from (F, \mathcal{R}) is non-terminating. ■

We exhibited a knowledge base where every exhaustive breadth-first **R**-derivation is infinite but there exists a terminating **R**-derivation. Now we will show a case where every exhaustive breadth-first **R**-derivation is terminating but there exists an infinite exhaustive **R**-derivation.

Example 23: Let $F = \{p(a, b)\}$ and \mathcal{R} the following set of rules:

$$R_1 = p(x, y) \rightarrow q(y)$$

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

$$R_3 = p(x, y) \rightarrow p(y, y)$$

By applying the rules in their order on $p(a, b)$ we obtain the expanded fact-base $\{p(a, b), q(b), p(b, z_{\mathbf{t}_2}), p(b, b)\}$. Now we can reapply the rules on the new atom $p(b, z_{\mathbf{t}_2})$ producing a similar expansion and this process can be repeated endlessly producing an infinite exhaustive **R**-derivation \mathcal{D} :

\emptyset	F	$Z_0 = F$
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{q(b)\}$	$Z_1 = F_1$
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto b\})$	$F_2 = F_1 \cup \{p(b, z_{\mathbf{t}_2})\}$	$Z_2 = F_2$
$\mathbf{t}_3 = (R_3, \{x \mapsto a, y \mapsto b\})$	$F_3 = F_2 \cup \{p(b, b)\}$	$Z_3 = F_3$
$\mathbf{t}_4 = (R_1, \{x \mapsto b, y \mapsto z_{\mathbf{t}_2}\})$	$F_4 = F_3 \cup \{q(z_{\mathbf{t}_2})\}$	$Z_4 = F_4$
$\mathbf{t}_5 = (R_2, \{x \mapsto b, y \mapsto z_{\mathbf{t}_2}\})$	$F_5 = F_4 \cup \{p(z_{\mathbf{t}_2}, z_{\mathbf{t}_5})\}$	$Z_5 = F_5$
$\mathbf{t}_6 = (R_3, \{x \mapsto b, y \mapsto z_{\mathbf{t}_2}\})$	$F_6 = F_5 \cup \{p(z_{\mathbf{t}_2}, z_{\mathbf{t}_2})\}$	$Z_6 = F_6$
\vdots	\vdots	\vdots

On the other hand, every breadth-first **R**-derivation from (F, \mathcal{R}) includes only the triggers \mathbf{t}_1 and \mathbf{t}_3 . That is because their application (in whatever order) leads to a factbase on which \mathbf{t}_2 is not **R**-applicable, guaranteeing termination.

Notice that the above derivation \mathcal{D} is also a **F**-derivation because there is no point where the (frugal) output of a trigger isomorphically subsumes a piece of the respective factbase. But \mathcal{D} is not a **V**-derivation, because in such a setting we would have to remove the atom $p(b, z_{\mathbf{t}_2})$ when we add $p(b, b)$. We can however create a similar example for the **V**-chase, showing that all exhaustive breadth-first **V**-derivations are terminating but there exists an infinite exhaustive **V**-derivation, if we modify our ruleset as follows:

$$R_1 = p(x, y) \rightarrow q(y)$$

$$R'_2 = p(x, y) \wedge q(y) \rightarrow \exists z p(y, z) \wedge r(y, z)$$

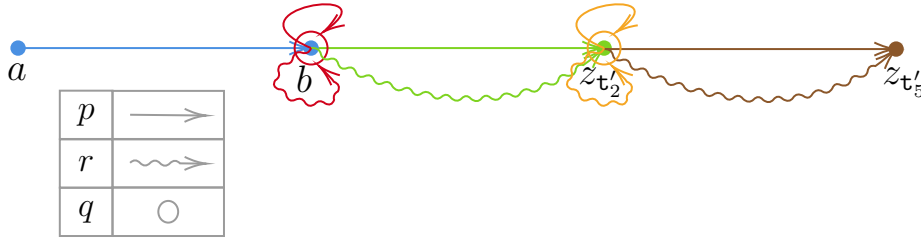
$$R_3 = p(x, y) \rightarrow p(y, y)$$

$$R_4 = p(x, y) \rightarrow r(y, y)$$

Let $\mathcal{R}' = \{R_1, R'_2, R_3, R_4\}$. Following a similar strategy like in \mathcal{D} , i.e. by applying the rules in the given order, we create an infinite exhaustive **V**-derivation $\mathcal{D}' = (\mathbf{t}'_*, F'_*, Z'_*)$ from (F, \mathcal{R}') :

\emptyset	F	$Z_0 = F$
$\mathfrak{t}'_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F'_1 = F \cup \{q(b)\}$	$Z'_1 = F'_1$
$\mathfrak{t}'_2 = (R'_2, \{x \mapsto a, y \mapsto b\})$	$F'_2 = F'_1 \cup \{p(b, z_{\mathfrak{t}'_2}), r(b, z_{\mathfrak{t}'_2})\}$	$Z'_2 = F'_2$
$\mathfrak{t}'_3 = (R_3, \{x \mapsto a, y \mapsto b\})$	$F'_3 = F'_2 \cup \{p(b, b)\}$	$Z'_3 = F'_3$
$\mathfrak{t}'_4 = (R_4, \{x \mapsto a, y \mapsto b\})$	$F'_4 = F'_3 \cup \{r(b, b)\}$	$Z'_4 = F'_4$
$\mathfrak{t}'_5 = (R_1, \{x \mapsto b, y \mapsto z_{\mathfrak{t}'_2}\})$	$F'_5 = F'_4 \cup \{q(z_{\mathfrak{t}'_2})\}$	$Z'_5 = F'_5$
$\mathfrak{t}'_6 = (R'_2, \{x \mapsto b, y \mapsto z_{\mathfrak{t}'_2}\})$	$F'_6 = F'_5 \cup \{p(z_{\mathfrak{t}'_2}, z_{\mathfrak{t}'_6}), r(z_{\mathfrak{t}'_2}, z_{\mathfrak{t}'_6})\}$	$Z'_6 = F'_6$
$\mathfrak{t}'_7 = (R_3, \{x \mapsto b, y \mapsto z_{\mathfrak{t}'_2}\})$	$F'_7 = F'_6 \cup \{p(z_{\mathfrak{t}'_2}, z_{\mathfrak{t}'_2})\}$	$Z'_7 = F'_7$
$\mathfrak{t}'_8 = (R_4, \{x \mapsto b, y \mapsto z_{\mathfrak{t}'_2}\})$	$F'_8 = F'_7 \cup \{r(z_{\mathfrak{t}'_2}, z_{\mathfrak{t}'_2})\}$	$Z'_8 = F'_8$
\vdots	\vdots	\vdots

Below we find the factbase Z'_8 , with the usual colors representing the ranks but notice that this time the derivation is not rank compatible, so the coloring does not correspond to the order with which the atoms are produced:



Although \mathcal{D}' is infinite and exhaustive, every exhaustive breadth-first V-derivation from (F, \mathcal{R}') is terminating since it will necessarily include the triggers $\mathfrak{t}'_1, \mathfrak{t}'_3$ and \mathfrak{t}'_4 , resulting in a factbase on which no trigger from \mathcal{R}' is R-applicable. ■

In the above example we showed that there exist knowledge bases where every exhaustive breadth-first F- or V-derivation is terminating but there exist infinite exhaustive F- and V-derivations. Now we will show a case where every exhaustive breadth-first F- or V-derivation is infinite but there exist terminating F- and V-derivations.

Example “FRUGGAMMA” 24: Let $F = \{r(a)\}$ and \mathcal{R} the following set of rules:

$$R_1 = q(x, v) \wedge p(x, y) \rightarrow \exists w p(x, w) \wedge t(w)$$

$$R_2 = r(x) \rightarrow \exists y p(x, y)$$

$$R_3 = r(x) \wedge p(x, y) \rightarrow \exists w q(x, w)$$

$$R_4 = p(x, y) \rightarrow \exists z p(y, z)$$

$$R_5 = t(x) \rightarrow p(x, x)$$

Here is a breadth-first **V**- and **F**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathbf{t}_1 = (R_2, \{x \mapsto a\})$	$F_1 = F \cup \{p(a, y_{\mathbf{t}_1})\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R_3, \{x \mapsto a, y \mapsto y_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{q(a, w_{\mathbf{t}_2})\}$	$Z_3 = F_3$	
$\mathbf{t}_3 = (R_4, \{x \mapsto a, y \mapsto y_{\mathbf{t}_1}\})$	$F_3 = F_2 \cup \{p(y_{\mathbf{t}_1}, z_{\mathbf{t}_3})\}$	$Z_2 = F_2$	2
$\mathbf{t}_4 = (R_1, \{x \mapsto a, v \mapsto w_{\mathbf{t}_2}, y \mapsto y_{\mathbf{t}_1}\})$	$F_4 = F_3 \cup \{p(a, w_{\mathbf{t}_4}), t(w_{\mathbf{t}_4})\}$	$Z_4 = F_4$	
$\mathbf{t}_5 = (R_4, \{x \mapsto y_{\mathbf{t}_1}, y \mapsto z_{\mathbf{t}_3}\})$	$F_5 = F_4 \cup \{p(z_{\mathbf{t}_3}, z_{\mathbf{t}_5})\}$	$Z_5 = F_5$	3
$\mathbf{t}_6 = (R_5, \{x \mapsto w_{\mathbf{t}_4}\})$	$F_6 = F_5 \cup \{p(w_{\mathbf{t}_4}, w_{\mathbf{t}_4})\}$	$Z_6 = F_6$	
$\mathbf{t}_7 = (R_4, \{x \mapsto z_{\mathbf{t}_3}, y \mapsto z_{\mathbf{t}_5}\})$	$F_7 = F_6 \cup \{p(z_{\mathbf{t}_5}, z_{\mathbf{t}_7})\}$	$Z_7 = F_7$	4
$\mathbf{t}_8 = (R_4, \{x \mapsto z_{\mathbf{t}_5}, y \mapsto z_{\mathbf{t}_7}\})$	$F_8 = F_7 \cup \{p(z_{\mathbf{t}_7}, z_{\mathbf{t}_8})\}$	$Z_8 = F_8$	5
\vdots	\vdots	\vdots	\vdots
$\mathbf{t}_n = (R_4, \{x \mapsto z_{\mathbf{t}_{n-2}}, y \mapsto z_{\mathbf{t}_{n-1}}\})$	$F_n = F_{n-1} \cup \{p(z_{\mathbf{t}_{n-1}}, z_{\mathbf{t}_n})\}$	$Z_n = F_n$	\vdots

In Figure 4.8 we see the active factbase of \mathcal{D} after the application of \mathbf{t}_8 . The curvy arc represents predicate q , whereas r is represented by a cycle and t by a square. The dotted lines serve to demonstrate how this kind of procedure can continue (infinitely). We notice that \mathcal{D} behaves exactly like a **R**-derivation, as no atoms were removed during the chase, but it is also an infinite exhaustive

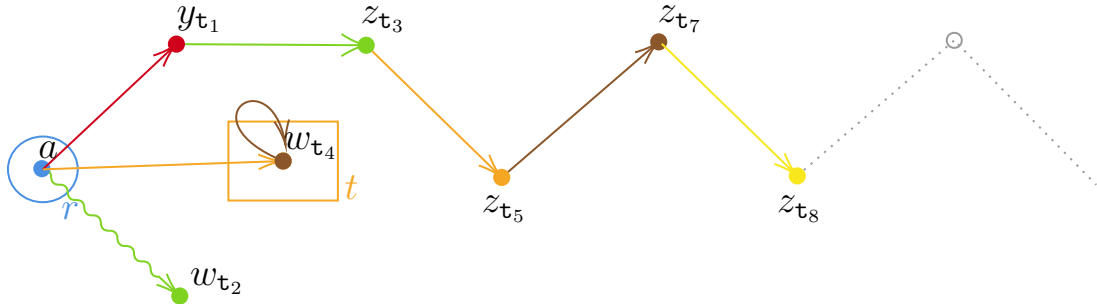


Figure 4.8: Example “FRUGGAMMA”, breadth-first \mathcal{D} : evolving Z_i .

F- and **V**-derivation. In addition, it is easy to verify that even if we changed the ordering of the triggers, by respecting the breadth-first property, we would not arrive at a terminating derivation. On the other hand, we have this **V**- and **F**-derivation from (F, \mathcal{R}) , which we call \mathcal{D}' :

\emptyset	F	$Z'_0 = F$	0
$\mathbf{t}_1 = (R_2, \{x \mapsto a\})$	$F'_1 = F \cup \{p(a, y_{\mathbf{t}_1})\}$	$Z'_1 = F'_1$	1
$\mathbf{t}_2 = (R_3, \{x \mapsto a, y \mapsto y_{\mathbf{t}_1}\})$	$F'_2 = F'_1 \cup \{q(a, w_{\mathbf{t}_2})\}$	$Z'_2 = F'_2$	2
$\mathbf{t}_4 = (R_1, \{x \mapsto a, v \mapsto w_{\mathbf{t}_2}, y \mapsto y_{\mathbf{t}_1}\})$	$F'_3 = F'_2 \cup \{p(a, w_{\mathbf{t}_4}), t(w_{\mathbf{t}_4})\}$	$Z'_3 = F'_3 \setminus \{p(a, y_{\mathbf{t}_1})\}$	3
$\mathbf{t}_6 = (R_5, \{x \mapsto w_{\mathbf{t}_4}\})$	$F'_4 = F'_3 \cup \{p(w_{\mathbf{t}_4}, w_{\mathbf{t}_4})\}$	$Z'_4 = F'_4$	4

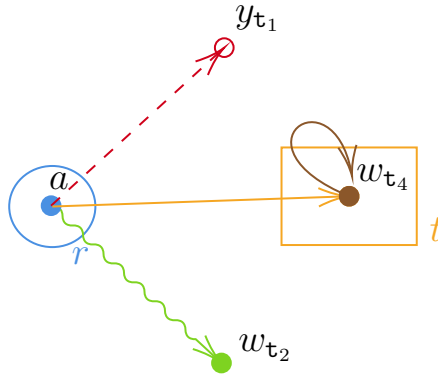


Figure 4.9: Example “FRUGGAMMA”, non-breadth-first \mathcal{D}' : factbase $F^{\mathcal{D}'}$.

In the above figure we see $F^{\mathcal{D}'}$, where the atom that is removed from $Z^{\mathcal{D}'}$ is shown with a dashed line. \mathcal{D}' is a terminating **F**- and **V**-derivation, as there is no trigger from \mathcal{R} that is **R**-applicable after the last rule application. ■

In Example 22, we saw that by confining our attention to breadth-first **R**-derivations from a given knowledge base, we might “miss” the existence of a possible terminating **R**-derivation. On the other hand, we contribute a positive result concerning the class of rank compatible **R**-derivations:

Proposition 4.7. For each terminating **R**-derivation from (F, \mathcal{R}) there exists a rank compatible terminating **R**-derivation from (F, \mathcal{R}) of smaller or equal depth.

Proof: Let \mathcal{D} be a terminating **R**-derivation from F and \mathcal{R} . Let $\text{trig}(\mathcal{D})$ be its sequence of associated triggers and let \mathcal{T} be a sorting of $\text{trig}(\mathcal{D})$ such that

the rank of each element is greater or equal to the rank of its predecessors. Let \mathcal{D}' be the derivation defined by applying, when **R**-applicable, the triggers using the order of \mathcal{T} . Because of the reordering, some of the triggers in \mathcal{T} may no longer be **R**-applicable in \mathcal{D}' . Nevertheless, \mathcal{D}' respects the rank compatibility property. We will show that it is a terminating **R**-derivation. Suppose that there is a new trigger $\mathfrak{t} \notin \mathcal{T}$ which is **R**-applicable on \mathcal{D}' , hence it is **R**-applicable on $F^{\mathcal{D}'}$. Then, since $F^{\mathcal{D}'} \subseteq F^{\mathcal{D}}$, we have that \mathfrak{t} is **O**-applicable on $F^{\mathcal{D}}$. But because \mathcal{D} is a terminating **R**-derivation, we know that \mathfrak{t} is not **R**-applicable on $F^{\mathcal{D}}$. Let $\mathfrak{t}_1, \dots, \mathfrak{t}_m$ be the triggers of $\text{trig}(\mathcal{D})$ that do not appear in \mathcal{T} . So

$$F^{\mathcal{D}} = F^{\mathcal{D}'} \cup \text{op}(\mathfrak{t}_1) \cup \dots \cup \text{op}(\mathfrak{t}_m) \quad (4.1)$$

Since \mathfrak{t} is not **R**-applicable on $F^{\mathcal{D}}$ we conclude that there is a substitution $\sigma : \text{nul}(\text{op}(\mathfrak{t})) \rightarrow \text{term}(F^{\mathcal{D}})$ such that $\sigma(\text{op}(\mathfrak{t})) \subseteq F^{\mathcal{D}}$. And since $\mathfrak{t}_1, \dots, \mathfrak{t}_m$ are not **R**-applicable in \mathcal{D}' we know that there are substitutions $\sigma_1, \dots, \sigma_m$ such that for every $i \in \{1, \dots, m\}$ we have $\sigma_i : \text{nul}(\text{op}(\mathfrak{t}_i)) \rightarrow \text{term}(F^{\mathcal{D}'})$ and $\sigma_i(\text{op}(\mathfrak{t}_i)) \subseteq F^{\mathcal{D}'}$. Since new variables are indexed by triggers, the domains of $\sigma_1, \dots, \sigma_m$ are pairwise disjoint and we can define the substitution $\dot{\sigma} = \bigcup_{i=1}^m \sigma_i$ which has the property that

$$\dot{\sigma}(F^{\mathcal{D}'} \cup \text{op}(\mathfrak{t}_1) \cup \dots \cup \text{op}(\mathfrak{t}_m)) = F^{\mathcal{D}'} \quad (4.2)$$

Moreover, the domain of $\dot{\sigma}$ is disjoint with the variable set $\text{var}(\text{sp}(\mathfrak{t}))$, because the new variables created from $\mathfrak{t}_1, \dots, \mathfrak{t}_m$ are not present in $F^{\mathcal{D}'}$. Therefore the composition $\dot{\sigma} \circ \sigma$ retains $\text{nul}(\text{op}(\mathfrak{t}))$ as its domain. So by 4.1 and $\sigma(\text{op}(\mathfrak{t})) \subseteq F^{\mathcal{D}}$ we can write

$$\dot{\sigma} \circ \sigma(\text{op}(\mathfrak{t})) \subseteq \dot{\sigma}(F^{\mathcal{D}'} \cup \text{op}(\mathfrak{t}_1) \cup \dots \cup \text{op}(\mathfrak{t}_m))$$

which with 4.2 becomes

$$\dot{\sigma} \circ \sigma(\text{op}(\mathfrak{t})) \subseteq F^{\mathcal{D}'}$$

so $F^{\mathcal{D}'}$ is a retract of $\dot{\sigma} \circ \sigma(\text{op}(\tau))$. This implies that τ is not **R**-applicable on \mathcal{D}' . That is a contradiction, which leads us to conclude that no such τ exists, therefore \mathcal{D}' is a terminating **R**-derivation. \square

In the above (counter-)examples we noticed that in the restricted, the frugal and the vacuum chase, i.e. when $X \in \{\mathbf{R}, \mathbf{F}, \mathbf{V}\}$, the behavior of breadth-first X -derivations towards termination is considerably different than the behavior of X -derivations. We conclude that breadth-first derivations do not suffice for studying the termination of the restricted, the frugal or the vacuum chase, unlike the case for any X -chase variant where $X \in \{\mathbf{O}, \mathbf{SO}, \mathbf{E}, \mathbf{P}, \mathbf{C}\}$. This implies that other strategies, different than the breadth-first approach, might provide good results for certain types of knowledge bases under certain types of chase variants. This is an important observation in itself, and it is not very evident.

Following this observation it is appropriate to consider the breadth-first approach to a chase variant which is not by definition breadth-first, as a different chase variant. After all, we are effectively restricting the class of derivations in consideration in such a way that changes the overall features of this class. Therefore we define:

Definition 4.21. Let X be a chase variant. The class of all breadth-first X -derivations is identified as the **bf- X -chase**. \dashv

By definition, for every chase variant X it holds that $\text{bf-}X \geq_t X$. Seen under this prism and considering Proposition 4.6 and Example 23, we summarize the relations of chase variants with their breadth-first sub-classes:

Proposition 4.8. The following relations concerning termination of (breadth-first) chase variants hold:

- $\text{bf-O} \equiv_t \mathbf{O}$,
- $\text{bf-SO} \equiv_t \mathbf{SO}$,
- $\text{bf-R} >_t \mathbf{R}$,
- $\text{bf-F} >_t \mathbf{F}$,

Notice that if we denote with **rc-R**-chase the class of all rank compatible **R**-derivations, Proposition 4.7 does not guarantee that the **rc-R**-chase is equivalent with the **R**-chase because the **R**-chase is termination-order dependent. We will not be exploring the **rc-R**-chase any further in this thesis, thus this question will be left open. However the relation between the **P**-chase and the **bf-R**-chase needs a clarification. By definition we can see that the **bf-R**-chase is as strong as the **P**-chase wrt termination. But what about the other direction? Are the two chase variants equivalent wrt termination? The following example shows that this is not the case.

Example 25: Let $F = \{p(a), p(b), p(c)\}$ and \mathcal{R} the following ruleset:

$$R_1 = p(x) \rightarrow \exists z \, q(z, z)$$

$$R_2 = q(x, x) \wedge q(y, y) \rightarrow \exists z \, q(x, z) \wedge q(z, x) \wedge q(z, z) \wedge q(y, z) \wedge q(z, y)$$

The triggers $\mathbf{t}_{1.1} = (R_1, \{x \mapsto a\})$, $\mathbf{t}_{1.2} = (R_1, \{x \mapsto b\})$ and $\mathbf{t}_{1.3} = (R_1, \{x \mapsto c\})$ are **R**-applicable on F . Every **bf-R**-derivation from (F, \mathcal{R}) is going to apply only one of the three adding the atom $q(z_{\mathbf{t}_1}, z_{\mathbf{t}_1})$ to the factbase (where $\mathbf{t}_1 \in \{\mathbf{t}_{1.1}, \mathbf{t}_{1.2}, \mathbf{t}_{1.3}\}$). After that there are no more triggers **R**-applicable to the factbase so the derivation is terminating at rank 1.

However all three $\mathbf{t}_{1.1}$, $\mathbf{t}_{1.2}$ and $\mathbf{t}_{1.3}$ will appear in every **P**-derivation from (F, \mathcal{R}) producing the atoms $q(z_{\mathbf{t}_{1.1}}, z_{\mathbf{t}_{1.1}})$, $q(z_{\mathbf{t}_{1.2}}, z_{\mathbf{t}_{1.2}})$ and $q(z_{\mathbf{t}_{1.3}}, z_{\mathbf{t}_{1.3}})$ at the first rank. This explodes the number of triggers **R**-applicable on the second rank. The reapplication of R_2 in factbases of more than 2 initial q -loops, leads to infinite **P**-derivations. Indeed the knowledge base $(\{q(c_1, c_1), q(c_2, c_2), q(c_3, c_3)\}, \{R_2\})$ does not even have a finite universal model. So every exhaustive **P**-derivation from (F, \mathcal{R}) is infinite. ■

Therefore the **bf-R**-chase is stronger than the **P**-chase wrt termination. In Figure 4.10 we provide a comprehensive table of the different chase variants and how they are related with respect to termination.

Much of our research has been done with the premise that the breadth-first strategy maintains an evident superiority in comparison to other choices of performing chase algorithms. Although there is arguably a lot of support for that

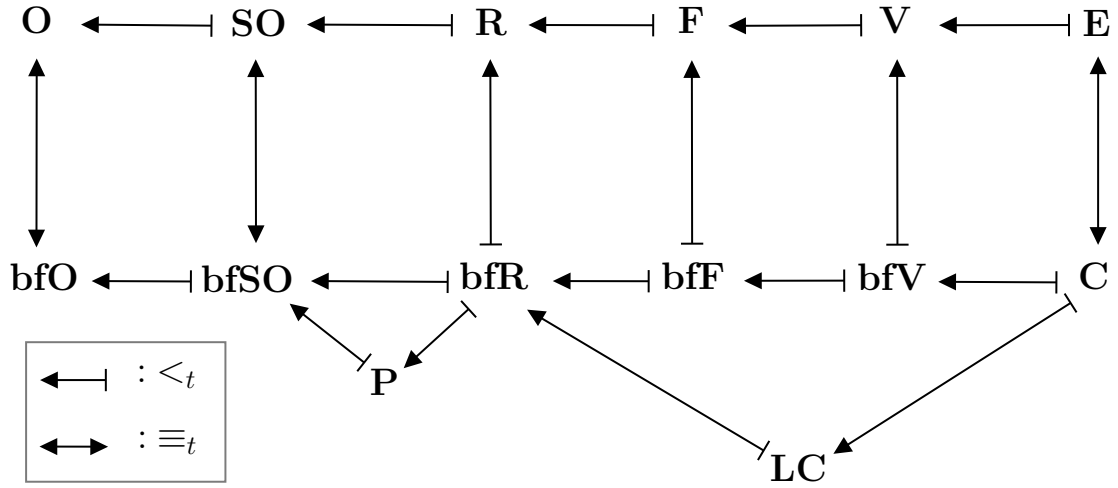


Figure 4.10: Chase variants and strength of termination. The LC-chase is going to be defined in Section 5.4.

claim, there are a number of elaborate examples, even from E- and C-chase, when the demand to perform all rule applications of a certain rank before continuing to the next rank, prevented redundancies from being detected and stalled the forward chaining process. We will provide relative examples later (see examples 37 and 31 in particular). We conclude that there is space for chase algorithms which might produce shorter terminating derivations, i.e. chase variants that are stronger in eliminating redundancy even from the C-chase (note that this refers to length and not depth of derivations, as C-derivations always are of optimal depth). Unfortunately such a quest is beyond the purpose of this study. Nonetheless, the findings outlined in this section are a step forward in our understanding of the potential of the forward chaining mechanism on (positive) existential rules.

4.5 Chase Graphs & Chase Space

In this section we provide some formal tools which can be useful when exploring properties related to derivations, as well as in visualizing particular examples of derivations. Towards the end of this section, several foundational properties are demonstrated, which will be crucial to some of the results presented in the next chapter.

4.5.1 Ancestors & Descendants

Before we introduce what is a *chase graph*, we proceed to discuss a notion that proves to be very useful in the context of derivations. This is the notion of *ancestors* and *descendants* in a derivation, which is employed in most of what follows in this thesis.

Definition 4.22 (Ancestors). Let \mathcal{D} be a derivation from (F, \mathcal{R}) and $A \in F^{\mathcal{D}}$, where A is produced¹² by τ . Then every atom in $\text{sp}(\tau)$ is called a *direct ancestor* of A in \mathcal{D} . The (indirect) *ancestor* relation between atoms is defined as the transitive closure of the direct ancestor relation, i.e. every direct ancestor of an atom A is an *ancestor* of A and if an atom A_1 is an ancestor of an atom A_2 and A_2 is an ancestor of an atom A_3 , then A_1 is also an *ancestor* of A_3 . We will represent the set of ancestors (in \mathcal{D}) of a set of atoms $Q \in F^{\mathcal{D}}$ as $\text{Anc}_{\mathcal{D}}(Q)$. The inverse of the (direct) ancestor relation is called the (*direct*) *descendant* relation, and the set of descendants of an atomset Q in \mathcal{D} is denoted with $\text{Desc}_{\mathcal{D}}(Q)$. Finally, the ancestors and the descendants of an atomset Q which are of rank i in \mathcal{D} are denoted with $\text{Anc}_{\mathcal{D}}^i(Q)$ and $\text{Desc}_{\mathcal{D}}^i(Q)$ respectively. \dashv

We accentuate that $A \in \text{op}(\tau)$ does not necessarily imply that the atoms of $\text{sp}(\tau)$ are the direct ancestors of A in every derivation \mathcal{D} with $\tau \in \text{trig}(\mathcal{D})$. That is because A might not be produced by τ in \mathcal{D} , instead it might be produced by some earlier trigger τ' , in which case the atoms of $\text{sp}(\tau')$ are the direct ancestors of A in \mathcal{D} . However, if A contains at least one variable from $\text{nul}(\text{op}(\tau))$, then we know that A belongs uniquely to $\text{op}(\tau)$ and is not found in the output of any other trigger. Therefore in this case, knowing that $\tau \in \text{trig}(\mathcal{D})$ ensures that A is produced by τ in \mathcal{D} , hence the direct ancestors of A in \mathcal{D} are veritably the atoms of $\text{sp}(\tau)$.

As we will see the ancestor relation facilitates a lot the analysis of derivations with respect to depth. Indeed, there is an evident correspondence between the notion of ancestors and the notions of rank and depth. Even more, the

¹²We remind that an atom A is *produced* by a trigger τ_i in a derivation $\mathcal{D} = (\tau_*, F_*, Z_*)$ if τ_i is the first trigger of $\text{trig}(\mathcal{D})$ with the property $A \in F_i \setminus Z_{i-1}$, i.e. only the part of the output of τ_i that has not been produced earlier in \mathcal{D} is indeed produced by τ_i in \mathcal{D} . This guarantees that each atom is produced by at most one trigger in a derivation.

ancestor/descendant relation is linked with **O**-applicability as shown in the following:

Lemma 4.2. Let \mathcal{D} be a derivation from (F, \mathcal{R}) and $A \in F^{\mathcal{D}}$. Let $F' = \text{Anc}_{\mathcal{D}}^0(A)$. Let \mathcal{T} be the subsequence¹³ of $\text{trig}(\mathcal{D})$ which includes all the triggers that produce an ancestor of A in \mathcal{D} , as well as the trigger that produces A . Then \mathcal{T} defines an **O**-derivation \mathcal{D}' from (F', \mathcal{R}) with $\text{trig}(\mathcal{D}') = \mathcal{T}$. Moreover $A \in F'^{\mathcal{D}'}$ and $\text{Anc}_{\mathcal{D}}(A) \subseteq F'^{\mathcal{D}'}$.

Proof: By induction on the number of the triggers in \mathcal{T} . If $\mathcal{T} = \emptyset$, then \mathcal{D}' is the trivial derivation from (F', \mathcal{R}) , including only the element $D'_0 = (\emptyset, F', F')$, which is indeed an **O**-derivation. We assume that the lemma holds for all trigger sequences \mathcal{T} with $|\mathcal{T}| = n$.

Let $\mathcal{T} = \mathfrak{t}_1, \dots, \mathfrak{t}_n, \mathfrak{t}_{n+1}$ be a subsequence of $\text{trig}(\mathcal{D})$ which comprises all the triggers that produce an ancestor of an atom $A \in F^{\mathcal{D}}$, including the trigger \mathfrak{t}_{n+1} that produces A . Then $\mathcal{T}' = \mathfrak{t}_1, \dots, \mathfrak{t}_n$ is the merging of all the sequences of triggers that produce the direct ancestors of A (respecting the ordering in $\text{trig}(\mathcal{D})$). Each one of those sequences is of length at most n , therefore it defines an **O**-derivation (by the induction hypothesis). In their merging \mathcal{T}' no trigger is repeated (since it is a merging), so \mathcal{T}' also defines an **O**-derivation \mathcal{D}'' from (F', \mathcal{R}) . But $F'^{\mathcal{D}''}$ includes all the ancestors of A in \mathcal{D} , therefore \mathfrak{t}_{n+1} is **O**-applicable on $F'^{\mathcal{D}''}$, producing the derivation \mathcal{D}' with $\text{trig}(\mathcal{D}') = \mathcal{T}$. Finally, $A \in F'^{\mathcal{D}'}$ and $\text{Anc}_{\mathcal{D}}(A) \subseteq F'^{\mathcal{D}'}$ are resulting from the fact that all triggers that produce these atoms in \mathcal{D} are present in \mathcal{D}' . \square

Every (finite) ruleset has a certain bound b to the number of atoms that appear in the rules' bodies. This implies that each atom produced with forward chaining has at most b direct ancestors. Furthermore, a chain of ancestors cannot exceed the depth of a derivation. These observations lead to the following lemma:

Lemma 4.3 (The ancestor clue). Let \mathcal{D} be a derivation from (F, \mathcal{R}) , where $\max\{|B| \mid (B, H) \in \mathcal{R}\} = b$. Then for any atom A of rank k in \mathcal{D} holds that $|\text{Anc}_{\mathcal{D}}^0(A)| \leq b^k$. \clubsuit

¹³A sequence S' is a *subsequence* of a sequence S if S' can be obtained from S by deleting some (or none) of the elements of S , without changing the order of the remaining elements.

Proof: We proceed by induction. If $\text{rank}(A) = 1$, then the ancestors of A of rank 0 are exactly the direct ancestors of A , so they are at most b . If the property holds for all atoms of rank up to $k - 1$, then an atom A of rank k will have at most b direct ancestors, which will all be of rank at most $k - 1$. And the zero-rank ancestors of A are exactly the zero-rank ancestors of all the direct ancestors of A , i.e. $\text{Anc}_{\mathcal{D}}^0(A) = \text{Anc}_{\mathcal{D}}^0(A_1) \cup \dots \cup \text{Anc}_{\mathcal{D}}^0(A_n)$ where A_1, \dots, A_n are the direct ancestors of A ($n \leq b$). So $|\text{Anc}_{\mathcal{D}}^0(A)| \leq b \cdot b^{k-1}$, which gives the requested $|\text{Anc}_{\mathcal{D}}^0(A)| \leq b^k$. \square

This lemma encompasses a simple idea. This idea, combined with the notion of *preservation of ancestry* which will be introduced in the next chapter, form the basis on which our results relating to boundedness are established. In short, from the above lemma we can easily verify that if an \mathbf{O} -derivation \mathcal{D} on a knowledge base (F, \mathcal{R}) is of depth k , then by tracking the ancestors of an atom of rank k in \mathcal{D} , we can construct an \mathbf{O} -derivation \mathcal{D}' from a knowledge base (F', \mathcal{R}) , where $F' \subseteq F$ and $|F'| \leq b^k$, with b being the maximum body size in \mathcal{R} . This bound in the size of the factbase that we need to consider to reach a certain depth will be a key to establishing the decidability of *k-boundedness* of a ruleset (to be introduced in the following chapter). The question is, how much can we exploit this strategy within a general chase framework? In other words, does this strategy work on all chase variants? If not then what are the characteristics that allow it to work? These questions will be answered but as we will see, when exiting the simple and clear waters of the \mathbf{O} -chase, a lot more detail has to be taken into account before arriving to concrete results.

4.5.2 Chase Graphs

From the concepts of the ancestor and the descendant relations between atoms, the idea of a chase graph emerges. This is a valuable tool for the study of the chase. Although the notion has appeared in the literature, it is formalized in ways which are not always equivalent and we have not based our definition on

any particular reference.¹⁴

Definition 4.23. Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ be a derivation from a knowledge base (F, \mathcal{R}) . Let $G = (V, E)$ be the directed acyclic graph where the nodes are atoms produced in \mathcal{D} and edges are added from an atom to its direct descendants labeled by the respective triggers. Formally,

- $V = F^{\mathcal{D}}$
- $E = \{(A_1, A_2) \mid A_2 \text{ is a direct descendant of } A_1 \text{ in } \mathcal{D}\}$ and
- there is a labeling function $\mathcal{L}_G : E \rightarrow \mathbf{trig}(\mathcal{D})$ where $\mathcal{L}_G((A_1, A_2)) = \mathfrak{t}$ if and only if A_2 is produced by \mathfrak{t} in \mathcal{D} and $A_1 \in \mathbf{sp}(\mathfrak{t})$.

G is said to be the *chase graph* associated with \mathcal{D} (or simply the *chase graph* of \mathcal{D}). A graph G that is associated with an X-derivation \mathcal{D} from (F, \mathcal{R}) is called an *X-chase graph* on (F, \mathcal{R}) . \dashv

To properly establish the notions of depth and rank in chase graphs we provide the following:

Proposition 4.9. Let $G = (V, E)$ be the chase graph of a derivation \mathcal{D} from (F, \mathcal{R}) . The rank of an atom $A \in F^{\mathcal{D}}$ in a derivation \mathcal{D} is equal to the maximum length of any path to this atom in the chase graph. The depth of \mathcal{D} is equal to the maximum length of any path in G .

Proof: We only need to prove that the rank of an atom $A \in F^{\mathcal{D}}$ is equal to the maximal length of any path to it in G , and the property about the depth then follows directly. We use induction. If $\mathit{rank}_{\mathcal{D}}(A) = 0$ then there is no path to A , which corresponds to a path of length 0. Now suppose that for every $A' \in F^{\mathcal{D}}$ with $\mathit{rank}_{\mathcal{D}}(A') \leq n - 1$ it holds that the maximal length of any path to this atom in G is $n - 1$. Let $A \in F_{\mathcal{D}}$ with $\mathit{rank}_{\mathcal{D}}(A) = n$. All the direct ancestors of A in \mathcal{D} are of rank at most $n - 1$, so they fulfill the induction hypothesis. There is at least one ancestor A' of A with $\mathit{rank}_{\mathcal{D}}(A') = n - 1$, so the longest

¹⁴There are cases where totally different notions have appeared with this name in the literature, like in [18] where “chase graph” is a graph that models dependencies between rules.

path to A' is of length $n - 1$. By adding one more edge (namely (A', A)) to this path, we obtain a path of length n to A . And we know that there is no longer path to A since all other ancestors have equal or smaller paths to them. So the proof is complete. \square

As a result we can say that the *rank* of an atom $A \in V$ in a chase graph $G = (V, E)$, denoted $\text{rank}_G(A)$, is equal to the maximum length of any path to A . The *depth* of G is the maximum length of any path in G . We will specifically frequently use the phrasing *k-deep (X-)chase graph G* , instead of (X-)chase graph G of depth k .

Example “GENERIC” 26, R-chase & C-chase Graph (continued from Examples 3, 4, 8 and 19): $F = \{p(a, b), p(c, d), r(e)\}$ and \mathcal{R} is the following set of rules:

$$\begin{aligned} R_1 &= p(x, y) \wedge r(z) \rightarrow p(y, z) \\ R_2 &= p(x, y) \wedge p(y, z) \rightarrow \exists u p(z, u) \\ R_3 &= p(x, y) \wedge p(x, z) \rightarrow p(y, z) \end{aligned}$$

We display a **R-derivation** \mathcal{D}'' from (F, \mathcal{R}) . Notice that we keep the same names for the triggers which were also used previously. As a consequence the order of the triggers does not follow the order of their index numbers.

\emptyset	F	$Z''_0 = F$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b, z \mapsto e\})$	$F''_1 = F \cup \{p(b, e)\}$	$Z''_1 = F''_1$	
$\mathbf{t}_4 = (R_1, \{x \mapsto c, y \mapsto d, y \mapsto e\})$	$F''_2 = F''_1 \cup \{p(d, e)\}$	$Z''_2 = F''_2$	
$\mathbf{t}_5 = (R_3, \{x \mapsto a, y \mapsto b, z \mapsto b\})$	$F''_3 = F''_2 \cup \{p(b, b)\}$	$Z''_3 = F''_3$	
$\mathbf{t}_6 = (R_3, \{x \mapsto c, y \mapsto d, z \mapsto d\})$	$F''_4 = F''_3 \cup \{p(d, d)\}$	$Z''_4 = F''_4$	1
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto b, z \mapsto e\})$	$F''_5 = F''_4 \cup \{p(e, u_{\mathbf{t}_2})\}$	$Z''_5 = F''_5$	
$\mathbf{t}_9 = (R_1, \{x \mapsto d, y \mapsto e, z \mapsto e\})$	$F''_6 = F''_5 \cup \{p(e, e)\}$	$Z''_6 = F''_6$	2
$\mathbf{t}_{10} = (R_3, \{x \mapsto e, y \mapsto u_{\mathbf{t}_2}, z \mapsto u_{\mathbf{t}_2}\})$	$F''_7 = F''_6 \cup \{p(u_{\mathbf{t}_2}, u_{\mathbf{t}_2})\}$	$Z''_7 = F''_7$	
$\mathbf{t}_{11} = (R_1, \{x \mapsto e, y \mapsto u_{\mathbf{t}_2}, z \mapsto e\})$	$F''_8 = F''_7 \cup \{p(u_{\mathbf{t}_2}, e)\}$	$Z''_8 = F''_8$	3

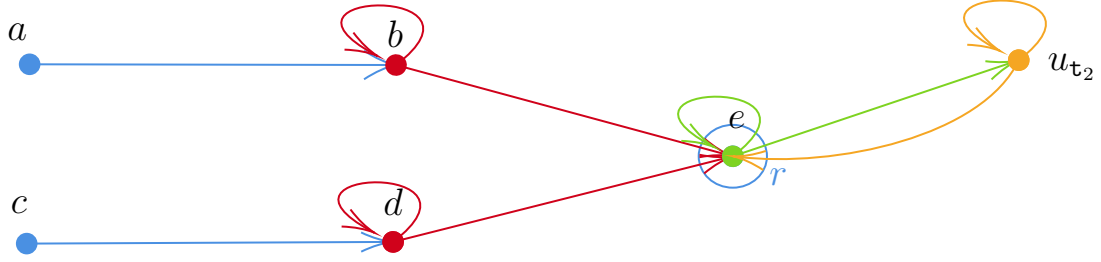


Figure 4.11: Example “GENERIC”, **R**-chase: representation of $F^{\mathcal{D}''}$.

\mathcal{D}'' is a terminating **bf-R**-derivation. The final factbase $F^{\mathcal{D}''}$ is shown in Figure 4.11, whereas below, in Figure 4.12, we find an illustration of the chase graph G'' of \mathcal{D}'' .

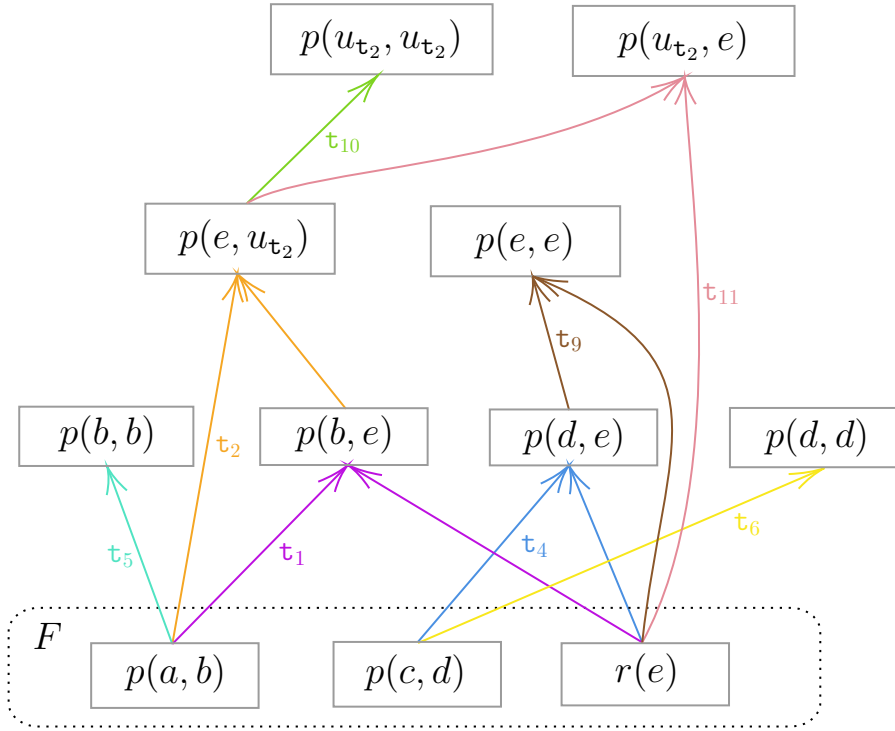


Figure 4.12: Example “GENERIC”, **R**-chase graph G'' of derivation \mathcal{D}'' .

In Section 4.3 (Example 19) we specified the **C**-derivation \mathcal{D}' on (F, \mathcal{R}) . It is interesting to compare the chase graph G' of \mathcal{D}' with G'' :

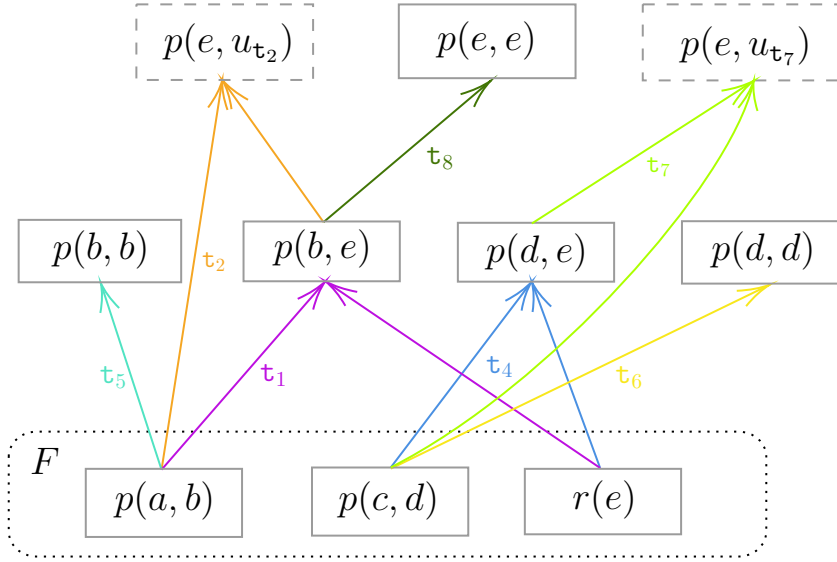


Figure 4.13: Example “GENERIC”, C-chase graph G' .

We used dashed boxes to indicate the two atoms that were produced but not included in the final active factbase $Z^{\mathcal{D}'}$. This information is not formally included in the chase graph. Notice that the atom $p(e, e)$ is produced in both derivations, but it is produced by different triggers. ■

The notions of ancestors and descendants can now be used in the context of chase graphs: let $G = (V, E)$ be a chase graph on (F, \mathcal{R}) . We will symbolize with V^i the set of all the atoms of rank i (where $i \in \{0, \dots, k\}$) in G . An atom $A \in V$ is an *ancestor* of an atom $A' \in V$ if there is a path from A to A' . And in this case A' is a *descendant* of A . And given $S \subseteq V$ we denote with $Anc_G(S)$ the set of all atoms which are ancestors of any atom of S in G . We also note with $Anc_G^i(S)$ the subset of $Anc_G(S)$ that contains only atoms of rank i , that is: $Anc_G^i(S) = Anc_G(S) \cap V^i$.

We highlight that every derivation specifies a unique chase graph, but each chase graph possibly corresponds to more than one derivations. If G is a chase graph, then we denote with $\mathbf{D}(G)$ the class of derivations corresponding to G , i.e. for every $\mathcal{D} \in \mathbf{D}(G)$ we have that G is associated with \mathcal{D} . These derivations can differ a lot in the order of the application of the triggers as well as at how the active factbase evolves. But they do have a common chase graph, so they

produce exactly the same set of atoms and the ranks of all atoms are the same, therefore they are also of the same depth.

Remark 4.4. Let $G = (V, E)$ be a chase graph and $\mathcal{D} \in \mathbf{D}(G)$. Then for all $A \in V$ holds that $\text{rank}_{\mathcal{D}}(A) = \text{rank}_G(A)$. ♣

In the previous example we compared a **R**-chase graph with a **C**-chase graph (both on the same knowledge base). Below we do the same with a **F**-chase graph and a **bf-F**-chase graph:

Example “FRUGGAMMA” 27, F-chase Graph (continued from Example 24):

Here we revisit an example from the previous section which served to show that in frugal chase, breadth-first derivations are not always optimal in terms of depth. We are working on the following knowledge base: $F = \{r(a)\}$ and \mathcal{R} is the following set of rules:

$$R_1 = q(x, v) \wedge p(x, y) \rightarrow \exists w p(x, w) \wedge t(w)$$

$$R_2 = r(x) \rightarrow \exists y p(x, y)$$

$$R_3 = r(x) \wedge p(x, y) \rightarrow \exists w q(x, w)$$

$$R_4 = p(x, y) \rightarrow p(y, z)$$

$$R_5 = t(x) \rightarrow p(x, x)$$

Below we depict the (**F**-)chase graphs G and G' , corresponding to derivations \mathcal{D} and \mathcal{D}' respectively. Actually G corresponds to a prefix of \mathcal{D} , since the derivation is infinite. In G' , we use a dashed box around $p(a, y_{t_1})$ to indicate that the atom is removed from the final active factbase.

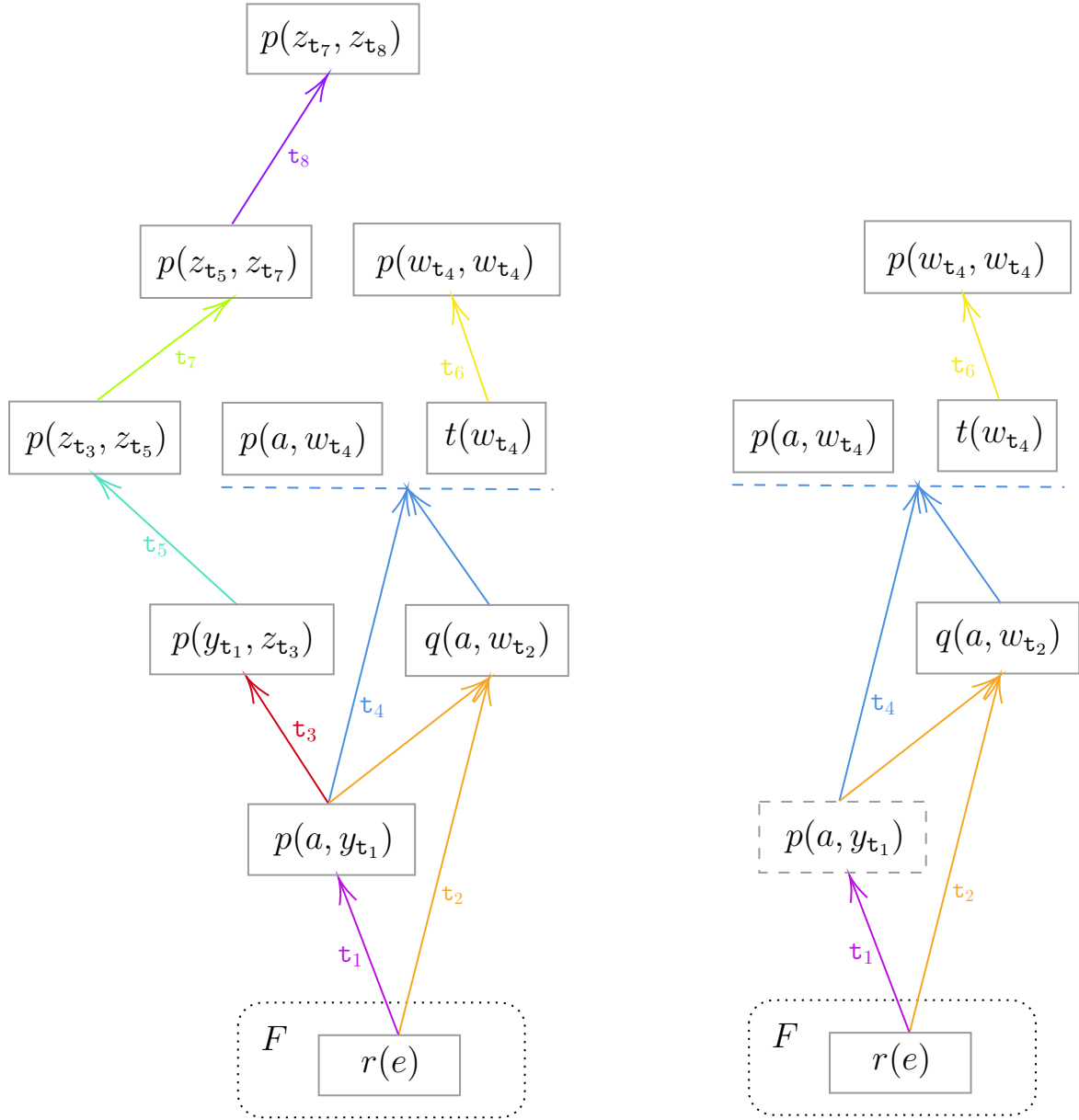


Figure 4.14: Example “FRUGGAMMA”, F-chase graphs G (left) and G' (right).

Notice that trigger τ_4 produces two atoms, thus formally there are edges to both of those atoms from their direct ancestors in G and G' . However it is convenient to represent the rule application with one edge per atom in the support of a trigger, and then connect the produced atoms with a dashed line as shown in the figure. This hints to yet another definition of a chase graph, with intermediate nodes to represent triggers (and connections from support to triggers and from triggers to produced atoms). For the purposes of this study it is best to formally handle the chase graph according to Definition 4.23 while using this visual representation. ■

Chase graphs can greatly facilitate our understanding of the mechanics of derivations. In this section we already saw two examples where in each we had two different derivations from the same knowledge base. From looking at the chase graphs in the last example, it is quite evident that in \mathcal{D}' we avoided a “path” of rule applications, and this contributed to obtaining a terminating \mathbf{F} -derivation. We already pointed out how this shows that there is a question of strategy when looking for a terminating \mathbf{X} -derivation from a certain knowledge base (in Section 4.4). But what is more important to stress here is how, given a particular knowledge base, there appears to be an underlying structure that defines the possible choices when deciding which rule applications to perform in forward chaining. In particular the “paths” available are always the same. Because the triggers available are always the same. The triggers are defined directly by the knowledge base. What we must do, is to find the right order to apply them.

4.5.3 The Chase Space

We saw how many relevant notions can be transferred from derivations to chase graphs. Hereafter we abstract a little more, defining a structure which represents all the possible choices involved in constructing a derivation from a particular knowledge base. The chase space corresponds to the union of all the possible chase graphs starting from a given knowledge base. It can be seen as a board on which, when performing forward chaining, we are choosing a path (actually a set of paths) based on specific rules depending on each chase variant. This path is directed from the atoms of rank 0 towards higher ranks.

The following remark leads us to the definition of the chase space:

Remark 4.5. Every knowledge base (F, \mathcal{R}) determines two critical sets:

- ▷ The set of all atoms that can be produced by a derivation from (F, \mathcal{R}) .
We denote this set with $F^{\mathcal{R}}$ and we can write

$$F^{\mathcal{R}} = \{A \in F^{\mathcal{D}} \mid \mathcal{D} \text{ is a derivation from } (F, \mathcal{R})\}$$

- ▷ The set of all triggers that can appear in a derivation from $F^{\mathcal{R}}$. We denote this set with \mathcal{R}^F and we can write

$$\mathcal{R}^F = \{\mathfrak{t} \in \text{trig}(\mathcal{D}) \mid \mathcal{D} \text{ is a derivation from } (F, \mathcal{R})\}$$

Thus for every $\mathfrak{t} \in \mathcal{R}^F$ it holds that $\text{sp}(\mathfrak{t}) \cup \text{op}(\mathfrak{t}) \subseteq F^{\mathcal{R}}$. ♣

Utilizing the above concepts, the definition of the chase space follows seamlessly:

Definition 4.24. Let (F, \mathcal{R}) be a knowledge base. The *chase space* of (F, \mathcal{R}) , denoted $\mathbb{C}(F, \mathcal{R})$, is a possibly infinite labelled directed graph whose set of nodes is $F^{\mathcal{R}}$ and whose set of edges is the set of pairs of atoms (A_1, A_2) such that there exists a trigger $\mathfrak{t} \in \mathcal{R}^F$ with $A_1 \in \text{sp}(\mathfrak{t})$ and $A_2 \in \text{op}(\mathfrak{t})$. In this case (A_1, A_2) is labelled with \mathfrak{t} . ⊥

The chase space can be used as a general framework to reason about forward chaining with positive existential rules. There is a clear connection between chase space and oblivious chase. And just like every derivation corresponds to an **O**-derivation with the same sequence of associated triggers (Remark 4.1¹⁵), we find that every chase graph is a subgraph of the respective chase space. In particular the following lemma is directly deduced:

Lemma 4.4. Let \mathcal{D} be an exhaustive **O**-derivation from (F, \mathcal{R}) . Then $F^{\mathcal{D}} = F^{\mathcal{R}}$ and $\text{trig}(\mathcal{D}) = \mathcal{R}^F$. More generally, if $G = (V, E)$ is an **X**-chase graph on (F, \mathcal{R}) , then G is a subgraph of $\mathbb{C}(F, \mathcal{R})$. ♣

In what follows we will correlate the chase space with the **bf-O**-chase specifically. But first, as a toy example of a chase space we revisit the Example 21:

Example “DATALOG” 28, Chase Space in Datalog (continued from Example 21): We have the factbase $F = \{p(a)\}$ and \mathcal{R} is the following set of rules:

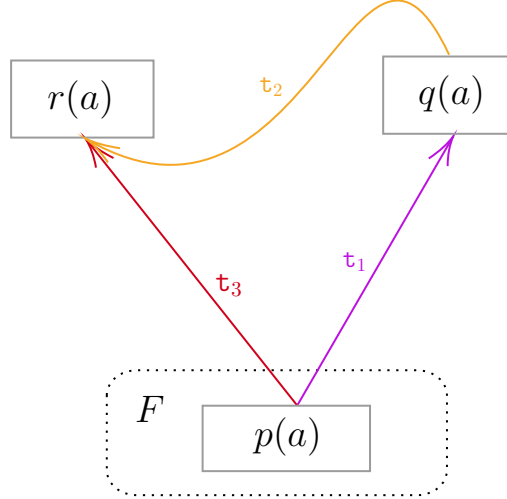
¹⁵Considering that every monotonic derivation is an **O**-derivation (and vice versa of course).

$$R_1 = p(x) \rightarrow q(x)$$

$$R_2 = q(x) \rightarrow r(x)$$

$$R_3 = p(x) \rightarrow r(x)$$

Below we find $\mathbb{C}(F, \mathcal{R})$:



As we can see, there are incoming edges to $r(a)$ labeled with different triggers. This signifies that there are different ways to obtain $r(a)$ with forward chaining in $\mathbb{C}(F, \mathcal{R})$. ■

At this point we clarify the main difference between chase graphs and chase spaces: in a chase graph we add an edge between atom A_1 and atom A_2 if A_2 is produced by a trigger that includes A_1 in its support. In a chase space we add an edge between atom A_1 and atom A_2 if A_2 can be produced by a trigger that includes A_1 in its support. On the other hand, in an exhaustive O-derivation \mathcal{D} from (F, \mathcal{R}) we necessarily apply all triggers of $\mathbb{C}(F, \mathcal{R})$. But the chase graph G that is associated with \mathcal{D} does not include any representation of triggers that do not produce atoms, nor does it represent the case when an atom appears in multiple trigger outputs (since the atom is only going to be produced once).

As with chase graphs, we can consider paths in a chase space as a way to pinpoint possible ancestors or possible descendants of an atom. To circumvent the fact that edges labeled with different triggers can now be directed to the same atom, we define the notion of *generator* of an atom A in a chase space

$\mathbb{C}(F, \mathcal{R})$ to be any maximal set of direct ancestors of A with the same label (i.e. the support of any trigger which includes A in its output). If an atom can be obtained by the application of different triggers in different derivations, then it will have more than one generator. In cases of non-terminating oblivious chase, some atoms can have an infinite number of generators. Here is a simple example:

Example 29: We have the factbase $F = \{p(a, b)\}$ and \mathcal{R} is the following set of rules:

$$R_1 = p(x, y) \rightarrow \exists z q(x, z)$$

$$R_2 = q(x, y) \rightarrow \exists z p(x, z)$$

$$R_3 = q(x, y) \rightarrow r(x)$$

In Figure 4.15 we can see an initial part of $\mathbb{C}(F, \mathcal{R})$ where

$$\begin{aligned} \mathbf{t}_1 &= (R_1, \{x \mapsto a, y \mapsto b\}) \\ \mathbf{t}_2 &= (R_2, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\}) \\ \mathbf{t}_3 &= (R_3, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\}) \\ \mathbf{t}_4 &= (R_1, \{x \mapsto a, y \mapsto z_{\mathbf{t}_2}\}) \\ \mathbf{t}_5 &= (R_2, \{x \mapsto a, y \mapsto z_{\mathbf{t}_4}\}) \\ \mathbf{t}_6 &= (R_3, \{x \mapsto a, y \mapsto z_{\mathbf{t}_4}\}) \\ \mathbf{t}_7 &= (R_1, \{x \mapsto a, y \mapsto z_{\mathbf{t}_6}\}) \\ \mathbf{t}_8 &= (R_2, \{x \mapsto a, y \mapsto z_{\mathbf{t}_7}\}) \\ &\vdots \\ &\vdots \end{aligned}$$

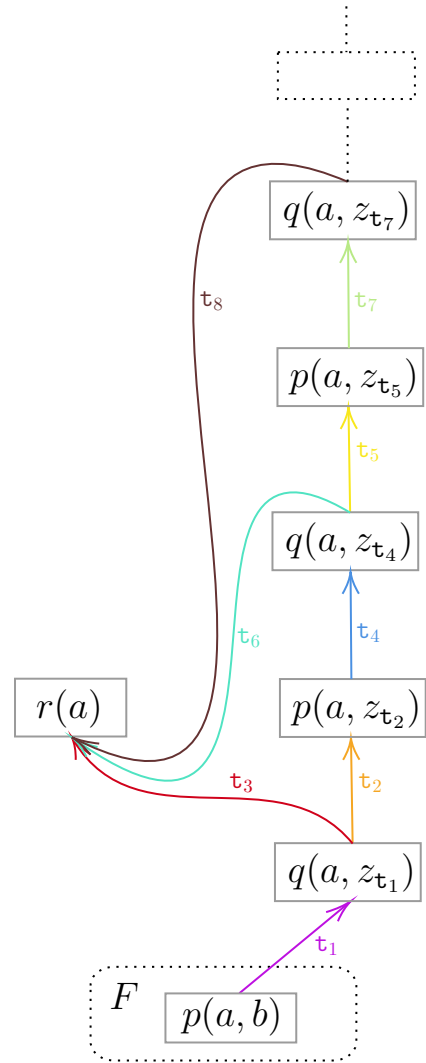


Figure 4.15: Example 29, chase space with infinite triggers having the same atom as output.

The chase space is infinite (hence not representable in entirety) but it is easy to see that there are infinite triggers that can potentially produce $r(a)$ in a derivation. In this case we can see that t_2 and all the triggers t_i with $i \geq 4$ produce redundant atoms and indeed, except for the oblivious chase, all other chase variants terminate after producing $q(a, z_{t_1})$ and $r(a)$. ■

After this discussion, we are faced with the question of specifying the rank of an atom in a chase space. A reasonable solution can be found following a simple observation:

Lemma 4.5. Let $\mathbb{C}(F, \mathcal{R})$ be a chase space, $A \in F^{\mathcal{R}}$ and $t \in \mathcal{R}^F$. If \mathcal{D} and \mathcal{D}' are bf-O-derivations from (F, \mathcal{R}) with $A \in F^{\mathcal{D}} \cap F^{\mathcal{D}'}$ and $t \in \text{trig}(\mathcal{D}) \cap \text{trig}(\mathcal{D}')$, then $\text{rank}_{\mathcal{D}}(A) = \text{rank}_{\mathcal{D}'}(A)$ and $\text{rank}_{\mathcal{D}}(t) = \text{rank}_{\mathcal{D}'}(t)$.

Proof: We can do induction on the rank, starting from the first rank, as there are no triggers of rank 0 anyway, and in both derivations, atoms of rank 0 are exactly the atoms of F .

Let $t \in \text{trig}(\mathcal{D})$ with $\text{rank}_{\mathcal{D}}(t) = 1$. Then $\text{sp}(t) \subseteq F$. So t will also be of rank 1 in \mathcal{D}' . Let $A \in F^{\mathcal{D}}$ and $\text{rank}_{\mathcal{D}}(A) = 1$. Then there is some $t \in \text{trig}(\mathcal{D})$ with $\text{rank}_{\mathcal{D}}(t) = 1$ such that $A \in \text{op}(t)$. Then, as we showed, it holds that $\text{rank}_{\mathcal{D}'}(t) = 1$ as well. So $\text{rank}_{\mathcal{D}'}(A) = 1$ (given that both derivations start from F and $A \notin F$).

Now suppose that for all $t \in \text{trig}(\mathcal{D})$ with $\text{rank}_{\mathcal{D}}(t) = n$ we have that $\text{rank}_{\mathcal{D}}(t) = \text{rank}_{\mathcal{D}'}(t)$ and for all $A \in F^{\mathcal{D}}$ with $\text{rank}_{\mathcal{D}}(A) = n$ we have that $\text{rank}_{\mathcal{D}}(A) = \text{rank}_{\mathcal{D}'}(A)$.

Let $t \in \text{trig}(\mathcal{D})$ with $\text{rank}_{\mathcal{D}}(t) = n + 1$. Since all atoms of $\text{sp}(t)$ are of rank at most n , they will have the same rank in \mathcal{D}' . In particular $\text{sp}(t)$ includes at least an atom A' of rank n in \mathcal{D} so $\text{rank}_{\mathcal{D}'}(A') = n$ as well. Hence $\text{rank}_{\mathcal{D}'}(t) = n + 1$.

Now let $A \in F^{\mathcal{D}}$ with $\text{rank}_{\mathcal{D}}(A) = n + 1$. Then it is produced by a trigger t with $\text{rank}_{\mathcal{D}}(t) = n + 1$. As we just showed, this implies that $\text{rank}_{\mathcal{D}'}(t) = n + 1$. As a result if A is produced by t in \mathcal{D}' , it will have $n + 1$ as its rank as well. If not then it is produced earlier. However it can't be produced in rank n , because

from the induction hypothesis, all atoms of rank n in \mathcal{D}' have the same rank in \mathcal{D} . So again A must be of rank $n + 1$ in \mathcal{D}' (even if it is not produced by τ).

The induction is complete and the statement is shown. \square

As a result we can provide the following definition:

Definition 4.25. Let $\mathbb{C}(F, \mathcal{R})$ be a chase space and let \mathcal{D} be a **bf-O**-derivation from (F, \mathcal{R}) . Then the *rank* of an atom $A \in F^{\mathcal{R}}$ in the chase space is specified as $\text{rank}_{(F, \mathcal{R})}(A) = \text{rank}_{\mathcal{D}}(A)$ and the *rank* of a trigger $\tau \in \mathcal{R}^F$ in $\mathbb{C}(F, \mathcal{R})$ is specified as $\text{rank}_{(F, \mathcal{R})}(\tau) = \text{rank}_{\mathcal{D}}(\tau)$. \dashv

In support of the above definition, we prove that the rank of an atom or a trigger in a chase space is the lower limit of the rank of this atom or trigger in any derivation from the corresponding knowledge base.

Proposition 4.10 (Minimum Rank). Let $\mathbb{C}(F, \mathcal{R})$ be a chase space and let \mathcal{D} be a derivation from (F, \mathcal{R}) . Then for every $A \in F^{\mathcal{D}}$ holds that $\text{rank}_{(F, \mathcal{R})}(A) \leq \text{rank}_{\mathcal{D}}(A)$.

Proof: Let \mathcal{D} be a derivation on (F, \mathcal{R}) . From Remark 4.1 we know that we can build an oblivious chase derivation \mathcal{D}' which will have exactly the same sequence of associated triggers. So the ranks of atoms in \mathcal{D} and \mathcal{D}' are the same. If \mathcal{D}' is not rank compatible, we can sort $\text{trig}(\mathcal{D}')$ so that it respects rank compatibility, creating another **O**-derivation \mathcal{D}'' . This is possible because by changing the order in this manner we do not affect the **O**-applicability of the triggers. We can envision this process as a step by step algorithm, where starting from the beginning of $\text{trig}(\mathcal{D}')$, each time we find a trigger of a lower rank appearing later in the sequence, we place it earlier, just after the rest of the triggers of its corresponding rank. As shown in Proposition 4.5, this transformation does not increase the ranks of the produced atoms. So, after a number of such operations we will obtain the rank compatible \mathcal{D}'' which produces exactly the same atoms with the same or lower ranks. If \mathcal{D}'' is not breadth-first, we need to apply some more triggers to complete the ranks, obtaining a **bf-O**-derivation \mathcal{D}''' (the application of these new triggers again does not influence **O**-applicability of the

rest). The addition of the new outputs earlier in the derivation, can again only decrease the ranks of already present atoms. Therefore from \mathcal{D} we obtained \mathcal{D}''' which is a **bf-O**-derivation, where each common atom has the same or lower rank. So for every $A \in F^{\mathcal{D}}$ holds that $\text{rank}_{(F, \mathcal{R})}(A) \leq \text{rank}_{\mathcal{D}}(A)$. \square

Before contributing a proposition concerning the case where we might reduce or expand a knowledge base, we accentuate that even with the same sequence of associated triggers, the ranks of a derivation can be affected by a change in the initial factbase. This is demonstrated by the following example, indicating that it is sensible to discern the two derivations as being different with different potential properties:

Example 30: Let $F = \{p(a), q(a)\}$ and \mathcal{R} be the following ruleset:

$$R_1 = p(x) \rightarrow q(x)$$

$$R_2 = p(x) \wedge q(x) \rightarrow r(x)$$

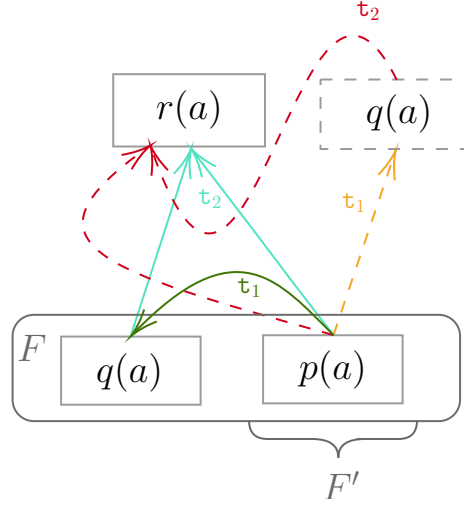
And here is an **O**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto a\})$	$F_1 = F$	$Z_1 = F_1$	
$\mathfrak{t}_2 = (R_2, \{x \mapsto a\})$	$F_2 = F_1 \cup \{r(a)\}$	$Z_2 = F_2$	1

Now if we take $F' = \{p(a)\}$ as the initial factbase, we can still apply the same triggers, obtaining \mathcal{D}' :

\emptyset	F'	$Z'_0 = F'$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto a\})$	$F'_1 = F'$	$Z'_1 = F'_1$	1
$\mathfrak{t}_2 = (R_2, \{x \mapsto a\})$	$F'_2 = F'_1 \cup \{r(a)\}$	$Z'_2 = F'_2$	2

Next, we illustrate a superposition of $\mathbb{C}(F, \mathcal{R})$ and $\mathbb{C}(F', \mathcal{R})$:



Notice that $\text{rank}_{\mathcal{D}}(\mathfrak{t}_2) = 2$ whereas $\text{rank}_{\mathcal{D}'}(\mathfrak{t}_2) = 1$. Correspondingly, $r(a)$ also has a different rank in the two derivations. ■

Altering the factbase is pertinent for the next chapter, where our main technique in order to arrive at certain results involves isolating a part of the factbase and applying the same ruleset to the reduced factbase. We can easily prove that reducing the knowledge base can only increase the ranks of the common triggers in the respective chase space.

Proposition 4.11 (Knowledge Base Reduction). Let $\mathbb{C}(F, \mathcal{R})$ and $\mathbb{C}(F', \mathcal{R}')$ be two chase spaces. If $F' \subseteq F$ and $\mathcal{R}' \subseteq \mathcal{R}$, then $\mathbb{C}(F', \mathcal{R}')$ is a subgraph of $\mathbb{C}(F, \mathcal{R})$. Moreover, for every $A \in F'^{\mathcal{R}'}$ it holds that $\text{rank}_{(F, \mathcal{R})}(A) \leq \text{rank}_{(F', \mathcal{R}')} (A)$.

Proof: The fact that $\mathbb{C}(F', \mathcal{R}')$ is a subgraph of $\mathbb{C}(F, \mathcal{R})$ is a consequence of the definition of chase space. In particular, every atom that can be produced from a knowledge base can surely be produced by a sub-knowledge base and consequently the same holds for the triggers that can appear in the derivations.

Now concerning the ranks, let $A \in F'^{\mathcal{R}'}$ and \mathcal{D}' be a **bf-O**-derivation from (F', \mathcal{R}') such that $A \in F'^{\mathcal{D}'}$. So $\text{rank}_{\mathcal{D}'}(A) = \text{rank}_{(F', \mathcal{R}')} (A)$. Then there is a derivation \mathcal{D} from (F, \mathcal{R}) with $\text{trig}(\mathcal{D}) = \text{trig}(\mathcal{D}')$. We will prove by induction that $\text{rank}_{\mathcal{D}}(A) \leq \text{rank}_{\mathcal{D}'}(A)$.

If $\text{rank}_{\mathcal{D}'}(A) = 0$ then $A \in F'$, so $A \in F$ hence $\text{rank}_{\mathcal{D}}(A) = 0$. If the property holds for every rank up to $i - 1$ and $\text{rank}_{\mathcal{D}'}(A) = i$,

then A is produced by a trigger \mathfrak{t} in \mathcal{D}' which has the property that $\max\{\text{rank}_{\mathcal{D}'}(A') \mid A' \in \text{sp}(\mathfrak{t})\} \leq i - 1$. So from the induction hypothesis we know also that $\max\{\text{rank}_{\mathcal{D}}(A') \mid A' \in \text{sp}(\mathfrak{t})\} \leq i - 1$, therefore $\text{rank}_{\mathcal{D}}(\mathfrak{t}) \leq \text{rank}_{\mathcal{D}'}(\mathfrak{t})$ and correspondingly $\text{rank}_{\mathcal{D}}(A) \leq \text{rank}_{\mathcal{D}'}(A)$, so the induction is complete.

Now from Proposition 4.10 we have that $\text{rank}_{\mathcal{D}}(A) \geq \text{rank}_{(F, \mathcal{R})}(A)$. As a result $\text{rank}_{(F', \mathcal{R}')} (A) \geq \text{rank}_{(F, \mathcal{R})}(A)$. \square

Following the above result, if a chase space \mathbb{C}_1 is a subgraph of another chase space \mathbb{C}_2 , we can say that \mathbb{C}_1 is a *subspace* of \mathbb{C}_2 .

Every derivation assigns ranks to its atoms and triggers, that are at least as big as the ranks of the chase space. We know that in **bf-O**-derivations (by definition) the ranks of atoms and triggers are equal to those of the chase space. But can we specify a bigger class of derivations which assigns ranks equal to those of the chase space? A reasonable candidate would be the class of all breadth-first chase variants. Interestingly and somewhat counter-intuitively, not every breadth-first chase variant assigns equal ranks to those of the chase space. The following (counter-)example concerns the equivalent chase.

Example 31: Let \mathcal{R} be the following ruleset:

$$R_1 = p(x) \rightarrow q(x)$$

$$R_2 = q(x) \rightarrow p(x)$$

$$R_3 = q(x) \rightarrow r(x)$$

$$R_4 = p(x) \rightarrow r(x)$$

$$R_5 = r(x) \rightarrow s(x)$$

$$R_6 = r(x) \rightarrow p(x)$$

and let F be the factbase $\{p(z_1), q(z_2)\}$, where z_1 and z_2 are variables. Then here is an **E**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto z_1\})$	$F_1 = F \cup \{q(z_1)\}$	$Z_1 = F_1$	
$\mathfrak{t}_2 = (R_3, \{x \mapsto z_2\})$	$F_2 = F_1 \cup \{r(z_2)\}$	$Z_2 = F_2$	
$\mathfrak{t}_3 = (R_4, \{x \mapsto z_1\})$	$F_3 = F_2 \cup \{r(z_1)\}$	$Z_3 = F_3$	1
$\mathfrak{t}_4 = (R_5, \{x \mapsto z_2\})$	$F_4 = F_3 \cup \{s(z_2)\}$	$Z_4 = F_4$	
$\mathfrak{t}_5 = (R_6, \{x \mapsto z_2\})$	$F_5 = F_4 \cup \{p(z_2)\}$	$Z_5 = F_5$	2

Notice that the atom $p(z_2)$ is of rank 2 in \mathcal{D} . But there is also the **E**-derivation \mathcal{D}' from (F, \mathcal{R}) :

\emptyset	F	$Z'_0 = F$	0
$\mathfrak{t}_6 = (R_2, \{x \mapsto z_2\})$	$F'_1 = F \cup \{p(z_2)\}$	$Z'_1 = F'_1$	
$\mathfrak{t}_2 = (R_3, \{x \mapsto z_2\})$	$F'_2 = F'_1 \cup \{r(z_2)\}$	$Z'_2 = F'_2$	1
$\mathfrak{t}_4 = (R_5, \{x \mapsto z_2\})$	$F'_3 = F'_2 \cup \{s(z_2)\}$	$Z'_3 = F'_3$	2

In derivation \mathcal{D}' , the rank of $p(z_2)$ is 1. Both \mathcal{D} and \mathcal{D}' are breadth-first, since anyway the **E**-chase is a breadth-first chase variant. Thus we have shown that different breadth-first X-chase derivations from the same knowledge base can attribute different ranks to the same atom. And we can even produce a similar counterexample where the initial factbase does not contain variables, by adding a rule that would produce a factbase isomorphic with F out of any initial (ground) atom. ■

In order to ensure the equality of ranks of atoms and triggers in X-derivations with their ranks in the respective chase spaces, we can specify a property which defines a much larger class than that of the **bf-O**-derivations. We will do so in the following chapter, proving that when $X \in \{\mathbf{O}, \mathbf{SO}, \mathbf{R}, \mathbf{P}\}$, every breadth-first X-derivation assigns ranks to its atoms and triggers which are equal to those of the chase space.

We have introduced the main theoretical tools that are needed to analyze depth-related properties of forward chaining with existential rules. We proceed

now to discuss the different manifestations of boundedness within this framework.

5 Characterizing Boundedness in Different Chase Variants

In this chapter we investigate how chase variants behave with respect to boundedness. In Section 1 we define boundedness and k -boundedness of a ruleset within a particular chase variant. In Section 2 we prove that k -boundedness is decidable for several chase variants, by demonstrating that they satisfy intermediate properties which then guarantee this decidability. In Section 3 we provide upper bounds for the computational complexity of determining k -boundedness in the cases where it is shown to be decidable. Then in Section 4 we define a new chase variant, the *local core chase*, which is stronger than the breadth-first restricted chase in eliminating redundancy while we believe that it retains the property that k -boundedness is decidable. In Section 5 we show the connection between boundedness and a certain kind of minimality of chase graphs and we discuss algorithms that generate these minimal chase graphs.

5.1 Boundedness & k -Boundedness

The main motivating problem in this thesis is how to decide whether a ruleset has an intrinsic structure which guarantees that every derivation with this ruleset has a bounded depth, independently of the factbase. This necessarily depends on the chase variant within which we operate. At the beginning of this section

the corresponding definitions are given and examples are discussed. Considering the hardness of solving boundedness without restricting the rules' classes (in fact, as we pointed out in Chapter 3, the general case is undecidable for all chase variants), most research considering notions similar to boundedness and termination, is focused on some particular rule classes. We, on the other hand, turn to the specification of k -boundedness, where the bound is given. This is advantageous, because, since we do not restrict the rule classes, we approach the boundedness problem from a different angle, where structural properties of the chase variants are coming to light. Therefore, we have a double gain, where the research around k -boundedness provides insight to various mechanics of chase variants. These will transpire in the following sections.

5.1.1 Boundedness

We consider the question of whether there is a way to predetermine the depth of X -derivations, especially when a particular ruleset is considered and the initial factbase can vary. This gives birth to the notion of boundedness parametrized by a chase variant, which we now define.

Definition 5.1. Let X be a chase variant. A ruleset \mathcal{R} is *X -bounded* if there is a $k \in \mathbb{N}$ such that for every factbase F , all X -derivations from (F, \mathcal{R}) are of depth at most k . \dashv

In Section 2.2 we provided a definition of *boundedness* based on the α -operator and logical entailment. It is clear that this concept of boundedness is equivalent with **C**-boundedness as well as **E**-boundedness.

It is useful to point out that X -boundedness is a property of a ruleset and not of a knowledge base. Moreover since for every finite X -derivation, there is an exhaustive X -derivation of equal or greater depth, and since every infinite derivation is of infinite depth, boundedness of all exhaustive X -derivations implies boundedness of all X -derivations. Hence, we do not need to explicitly include a separate definition with the requirement that the bound refers to exhaustive X -derivations. The fact that boundedness has to be “tested” against all possible derivations from all possible factbases, makes it a very strong property.

On the other hand, a single infinite X-derivation from any factbase suffices to show that a ruleset is not X-bounded. Indeed, a corollary of the above definition is that if a ruleset \mathcal{R} is X-bounded, then every exhaustive X-derivation with \mathcal{R} , is a terminating X-derivation. Nevertheless it is not at all straightforward to determine whether a ruleset is bounded.

Example “GENERIC” 32, X-boundedness (continued from Examples 3, 4, 8, 19 and 26): We re-utilize the ruleset \mathcal{R} but we rename the variables of R_3 for convenience:

$$R_1 = p(x, y) \wedge r(z) \rightarrow p(y, z)$$

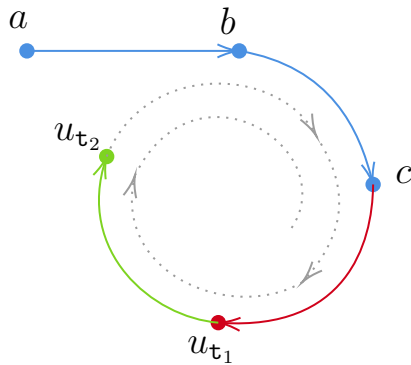
$$R_2 = p(x, y) \wedge p(y, z) \rightarrow \exists u p(z, u)$$

$$R_3 = p(\bar{x}, \bar{y}) \wedge p(\bar{x}, \bar{z}) \rightarrow p(\bar{y}, \bar{z})$$

To show that \mathcal{R} is not **R**-bounded, consider $F = \{p(a, b), p(b, c)\}$ and the derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathbf{t}_1 = (R_2, \{x \mapsto a, y \mapsto b, z \mapsto c\})$	$F_1 = F \cup \{p(c, u_{\mathbf{t}_1})\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R_2, \{x \mapsto b, y \mapsto c, y \mapsto u_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{p(u_{\mathbf{t}_1}, u_{\mathbf{t}_2})\}$	$Z_2 = F_2$	3
$\mathbf{t}_3 = (R_2, \{x \mapsto c, y \mapsto u_{\mathbf{t}_1}, y \mapsto u_{\mathbf{t}_2}\})$	$F_3 = F_2 \cup \{p(u_{\mathbf{t}_2}, u_{\mathbf{t}_3})\}$	$Z_3 = F_3$	4
\vdots	\vdots	\vdots	
\vdots	\vdots	\vdots	

Here is how the active factbase of \mathcal{D} evolves:



\mathcal{D} is an infinite **R**-derivation from (F, \mathcal{R}) , thus \mathcal{R} is not **R**-bounded. And every **R**-derivation is also an **O**- and **SO**-derivation so we conclude that \mathcal{R} is neither

SO-bounded nor O-bounded. Furthermore, \mathcal{D} is also a **F**- and a **V**-derivation. Hence \mathcal{R} is also not **F**-bounded neither **V**-bounded.

The only rule that creates new variables is R_2 . Let (R_2, π) be a trigger. Notice that the application of $(R_3, \bar{\pi})$ before (R_2, π) , where $\bar{\pi}(\bar{x}) = \pi(y)$ and $\bar{\pi}(\bar{y}) = \bar{\pi}(\bar{z}) = \pi(z)$, renders $\text{op}((R_2, \pi))$ redundant therefore the **E**-chase and the **C**-chase (which operate in a breadth-first manner) will detect this redundancy and future triggers will not include the output of (R_2, π) in their support. We conclude that \mathcal{R} is **C**-bounded and **E**-bounded. ■

In the following it will frequently be easier to reason on X-chase graphs, instead of X-derivations. Consequently we provide the following immediate proposition which is basically a reformulation of the definition 5.1 in terms of X-chase graphs.

Proposition 5.1. A ruleset \mathcal{R} is X-bounded if there exists a $k \in \mathbb{N}$ such that for every factbase F , every X-chase graph on (F, \mathcal{R}) is of depth at most k . ♣

As already mentioned, boundedness is known to be undecidable for classes of existential rules like Datalog (see Chapter 3). And it is undecidable even for a single ternary Datalog recursive rule. This indicates that the X-boundedness problem is going to be hard to solve even for relatively simple classes of existential rules. Can we identify recognizable classes of existential rules where X-boundedness is decidable? An affirmative answer to this question is given with the class of linear rules¹ for some chase variants. We will elaborate on this point in the following section.

5.1.2 k -Boundedness

We explained how boundedness is probably undecidable for various interesting classes of existential rules (since we already know that it is undecidable every time a “single ternary recursive datalog rule” is permitted). However, the practical interest of this notion lies more on whether we can find the particular bound k , rather than knowing that there exists one and thus the ruleset is

¹As specified in earlier chapters, *linear* rules are those whose bodies consist of one single atom.

bounded. Because even if we cannot know whether a ruleset is bounded or not, it can be useful to be able to check a particular bound k . To this aim, we define the notion of k -boundedness where the bound k is predefined.

Definition 5.2 (k -boundedness). Given a chase variant X , a ruleset \mathcal{R} is X - k -bounded if for every factbase F , any X -derivation from (F, \mathcal{R}) is of depth at most k . \dashv

Hence, in terms of chase graphs, we can say that a ruleset \mathcal{R} is X - k -bounded if for every factbase F , every X -chase graph on (F, \mathcal{R}) is of depth at most k . Note that a ruleset which is k -bounded is also bounded, but the converse is not true. When we discuss about the general concept of X - k -boundedness without needing to specify any particular chase variant X , we will simply refer to k -boundedness.

Example 33: Suppose we have the ruleset \mathcal{R} :

$$R_1 = p(x, y) \rightarrow \exists z q(z, x)$$

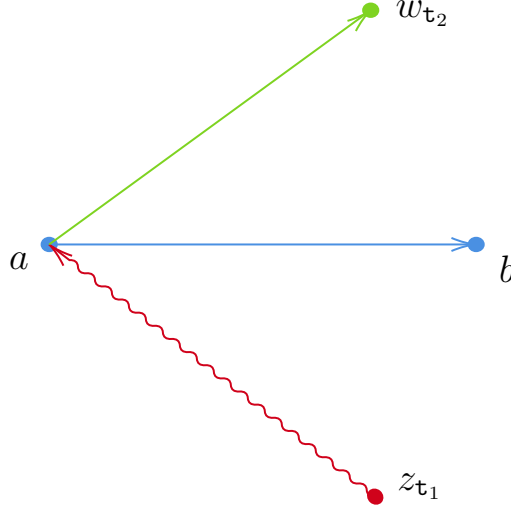
$$R_2 = q(z, x) \rightarrow \exists w p(x, w)$$

Since the rules are linear, atomic factbases suffice in order to test k -boundedness². Moreover, all initial factbases result to similar derivations because of the symmetric form of the ruleset. By repeating applications of R_1 and R_2 on an initial fact $p(a, b)$ we arrive at an infinite **O**-derivation, concluding that \mathcal{R} is not **O**-bounded. On the other hand the **SO**-chase halts at depth 2 as we can see in the following **SO**-derivation \mathcal{D} from $(\{p(a, b)\}, \mathcal{R})$:

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{q(z_{\mathbf{t}_1}, a)\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R_2, \{x \mapsto a, z \mapsto z_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{p(a, w_{\mathbf{t}_2})\}$	$Z_2 = F_2$	2

In the following graph, q is represented with the curvy arrow:

²This is formally established for the **O**-, **SO**- and **R**-chase with Proposition 5.2 in the next section, showing that in linear rules the (atomic factbase) chase termination is equivalent with boundedness.



We conclude that \mathcal{R} is 2-SO-bounded, however it is only 1-R-bounded: indeed t_2 is not R-applicable on Z_1 and the result is similar if we use $p(a, a)$, $q(a, a)$ or $q(a, b)$ as the starting factbase. Consequently it is also 1-F-bounded, 1-V-bounded, 1-C-bounded and 1-E-bounded. ■

5.2 Preservation of Ancestry

In this section, we identify a common property that allows us to prove that k -X-boundedness is decidable for $X \in \{\mathbf{O}, \mathbf{bf-O}, \mathbf{SO}, \mathbf{bf-SO}, \mathbf{R}, \mathbf{bf-R}, \mathbf{P}\}$. This common property, called *preservation of ancestry*, ensures that we can limit the size of the factbases where we need to test X- k -boundedness, which guarantees the decidability of the problem. However, we do not prove directly that all these chase variants preserve ancestry. Instead, we basically split those chase variants into two categories, and we show that they satisfy intermediate properties, which then imply preservation of ancestry, hence also decidability of k -boundedness.

The section consists of four parts:

- 1) We define *preservation of ancestry*, we prove that it implies decidability of k -boundedness and we also prove that it implies the equivalence of all-factbase termination with boundedness in the specific case of linear rulesets.

- 2) We define *heredity*, we show that the O-, bf-O-, SO- and R-chase variants are hereditary and then we prove that heredity implies preservation of ancestry.
- 3) We define **bf-R-compliance** and show that bf-R-compliant chase variants enjoy some convenient properties, which allow us to prove that the bf-SO-, bf-R- and P-chase variants preserve ancestry.
- 4) We show that the F-, V-, E- and C-chase do not preserve ancestry.

5.2.1 The Link with k -Boundedness

Here we specify the property which, when satisfied by a chase variant X, implies that we can restrict our attention to a finite number of factbases when testing X- k -boundedness of a particular ruleset \mathcal{R} .

Definition 5.3 (Ancestry). The X-chase is said to *preserve ancestry* if, for every X-derivation \mathcal{D} from (F, \mathcal{R}) , for every atom A in $F^{\mathcal{D}}$, there exists an X-derivation \mathcal{D}' from $(Anc_{\mathcal{D}}^0(A), \mathcal{R})$ such that A is produced in \mathcal{D}' and $rank_{\mathcal{D}}(A) = rank_{\mathcal{D}'}(A)$. \dashv

It is rather evident that we want to use the notion of ancestry to bound the size of the factbases that have to be considered, when investigating k -boundedness. This can be achieved considering the “ancestor clue” (Lemma 4.3). Since the number of ancestors of an atom of a certain rank is bounded (ancestor clue), if only the ancestors suffice to produce an atom at a certain rank (preservation of ancestry), then to know if a ruleset can produce atoms with a rank higher than k (k -boundedness), we only need to test derivations on factbases of a bounded size.

Theorem 5.1. Determining if a set of rules is X- k -bounded is decidable if the X-chase preserves ancestry.

Proof: Let X be a chase variant that preserves ancestry. Let \mathcal{R} be a ruleset. Suppose that \mathcal{R} is not X- k -bounded. Therefore there is a factbase F and a

derivation \mathcal{D} from (F, \mathcal{R}) with depth more than k . So there exists an atom $A \in F^{\mathcal{D}}$ with $\text{rank}(A) = k + 1$. Because the X-chase preserves ancestry, there exists an X-derivation \mathcal{D}' from $(\text{Anc}_{\mathcal{D}}^0(A), \mathcal{R})$ which produces A with the same rank as \mathcal{D} . Therefore \mathcal{D}' is also of depth more than k . Let b be the maximum number of atoms in the bodies of the rules of \mathcal{R} . By Lemma 4.3, we know that $\text{Anc}_{\mathcal{D}}^0$ has at most b^{k+1} atoms. We deduce that if a ruleset \mathcal{R} is not X- k -bounded, then there exists a factbase F' of at most b^{k+1} (where b depends on \mathcal{R}) such that there is an X-derivation from (F', \mathcal{R}) of depth more than k . The inverse of this statement is trivially true. In conclusion, if X is a chase variant that preserves ancestry, then a ruleset \mathcal{R} (with b maximum body size) is X- k -bounded if and only if for every factbase F' of size at most b^{k+1} , every X-derivation from (F', \mathcal{R}) is of depth at most k . Up to quasi-isomorphism, there is a finite number of factbases smaller than b^{k+1} . Let \mathcal{F} be the set of all factbases of size at most b^{k+1} , i.e. \mathcal{F} includes a representative of every quasi-equivalence class of factbases of size at most b^{k+1} . For every $F \in \mathcal{F}$, there is a finite number of X-derivations from (F, \mathcal{R}) with depth at most $k + 1$. Hence we can do the following:

- for each $F \in \mathcal{F}$, compute all X-derivations from (F, \mathcal{R}) with depth at most $k + 1$.
- if at least one such X-derivation has depth $k + 1$, then \mathcal{R} is not X- k -bounded.
- if all such X-derivations have depth at most k , then \mathcal{R} is X- k -bounded.

So we have shown that there exists a sound and complete way to verify whether \mathcal{R} is X- k -bounded or not. □

The preceding theorem suggests investigating preservation of ancestry in order to assure the decidability of k -boundedness. This research involves starting from a derivation on a knowledge base, then reducing the factbase and trying to reproduce an atom in the same rank. That makes the notion of chase space very pertinent, as we will be comparing different derivations on similar knowledge bases. Hence the results of Section 4.5 will be valuable in keeping track of the

changing of ranks of common atoms when we switch from one derivation to another, or from one knowledge base to a smaller one.

Before investigating which of the known chase variants preserve ancestry, we show how this property can also lead to results related to boundedness for linear rulesets. A ruleset is called *linear* if every one of its rules has a single atom body. We prove the following:

Proposition 5.2. Let X be a chase variant that preserves ancestry. A linear ruleset \mathcal{R} is X -bounded if and only if it has the property that for any factbase F , every exhaustive X -derivation from (F, \mathcal{R}) is terminating.

Proof: (\Rightarrow ;) Let \mathcal{R} be a set of linear existential rules. If \mathcal{R} is X -bounded then every X -derivation with \mathcal{R} has a bounded depth, and therefore cannot be infinite. Hence there is no infinite exhaustive X -derivation with \mathcal{R} , so for every factbase F , every exhaustive derivation from (F, \mathcal{R}) is terminating.

(\Leftarrow ;) Let \mathcal{R} be such that for any factbase F , every exhaustive X -derivation from (F, \mathcal{R}) is terminating. We define the *critical instance* to be a collection \mathcal{F} of representatives of the quasi-equivalence classes of all the atomic factbases whose atom has any predicate that appears in \mathcal{R} . Hence \mathcal{F} is finite. We will show that the maximum depth of an X -derivation from (F', \mathcal{R}) where $F' \in \mathcal{F}$, is equal to the maximum depth of any X -derivation with \mathcal{R} (independently of the factbase).

To show this we chose a random X -derivation \mathcal{D} from (F, \mathcal{R}) , where F is any factbase. Let k be the depth of \mathcal{D} . We will show that there exists an X -derivation \mathcal{D}' from (F', \mathcal{R}) , where $F' \in \mathcal{F}$ with depth $k' \geq k$. Let A be an atom that is produced at rank k in \mathcal{D} . Let T_A be the subsequence of $\text{trig}(\mathcal{D})$ that contains all the triggers that produced any ancestor of A in \mathcal{D} , as well as the trigger that produced A in \mathcal{D} .

We know that X preserves ancestry. So there exists an X -derivation \mathcal{D}'' from $\text{Anc}_{\mathcal{D}}^0(A)$ which produces A at the same rank as \mathcal{D} , therefore \mathcal{D}'' is of depth at least k . And because \mathcal{R} is a linear ruleset, we know that $\text{Anc}_{\mathcal{D}}^0(A)$ is a singleton set, i.e. it includes only one atom. Hence $\text{Anc}_{\mathcal{D}}^0(A)$ is quasi-equivalent to a representative $F' \in \mathcal{F}$. Therefore we can construct an X -derivation \mathcal{D}'

from (F', \mathcal{R}) with depth $k' \geq k$. □

As we mentioned in Section 3.2, for the class of linear rules it has been shown that given a ruleset, the all-factbase termination³ of the **O**- and **SO**-chase is decidable. This implies that the cases of the **bf-O**- and **bf-SO**-chase are also decidable. Moreover in extra linear rules (single atom head), it has been shown that the **R**-chase all-factbase termination is decidable which implies that the **bf-R**- and **P**-chase all-factbase terminations are also decidable. In this section we will show that the **X**-chase preserves ancestry when $X \in \{\mathbf{O}, \mathbf{bf-O}, \mathbf{SO}, \mathbf{bf-SO}, \mathbf{P}, \mathbf{R}, \mathbf{bf-R}\}$. Therefore, given a linear ruleset, the problem of determining whether it is **X**-bounded for $X \in \{\mathbf{O}, \mathbf{bf-O}, \mathbf{SO}, \mathbf{bf-SO}\}$ is decidable, whereas if the ruleset is extra linear, **X**-boundedness for $X \in \{\mathbf{O}, \mathbf{bf-O}, \mathbf{SO}, \mathbf{bf-SO}, \mathbf{P}, \mathbf{R}, \mathbf{bf-R}\}$ is decidable.

The next step is to prove that the aforementioned chase variants preserve ancestry. We will employ two different methods by identifying two independent classes of chase variants, which will be used to show that those chase variants preserve ancestry.

5.2.2 Heredity

The preservation of ancestry can be regarded as a top-down approach to identifying the part of a derivation that contributes to producing a certain atom. We find the atom, we trace down its ancestors and we effectively keep the whole structure in a new derivation which produces the same atom at the same rank from a smaller factbase. We proved that if we can achieve this, then k -boundedness is decidable for this chase variant.

But to show that this is possible in some concrete chase variants, we will actually use an inverse approach (bottom-up), where we show that by selecting a subset of the factbase, we can reproduce the part of the derivation that stems from this subset. To this end, we single out the triggers whose support is in the sub-factbase, or is produced only by that sub-factbase.

³The appellations “chase termination” and “all-instance termination” are fairly common, but with our terminology we should say all-factbase termination to mean “for any factbase F , every exhaustive **X**-derivation from (F, \mathcal{R}) is terminating”.

Definition 5.4 (Restriction induced by sub-factbase). A sequence of triggers \mathcal{T} is *applicable* on a knowledge base (F, \mathcal{R}) if there exists a derivation \mathcal{D} from (F, \mathcal{R}) with $\text{trig}(\mathcal{D}) = \mathcal{T}$. Now let \mathcal{D} be a derivation from (F, \mathcal{R}) and $F' \subseteq F$. The maximal subsequence of $\text{trig}(\mathcal{D})$ which is applicable on (F', \mathcal{R}) is called the *restriction* of $\text{trig}(\mathcal{D})$ induced by F' . \dashv

Note that in the limit case where $\mathcal{T} = \emptyset$, we can say that \mathcal{T} is applicable on any derivation $\mathcal{D} = D_0$, with $D_0 = (\emptyset, F, Z)$ (where F is any factbase). Similarly, the restriction of $\text{trig}(\mathcal{D})$ can be empty if it is induced by a subset that does not include the support of any trigger in \mathcal{D} .

Let \mathcal{D} be a derivation from (F, \mathcal{R}) and $F' \subseteq F$. Since the restriction \mathcal{T} of $\text{trig}(\mathcal{D})$ induced by F' is an applicable (on F') sequence of triggers, this implies that there exists a derivation \mathcal{D}' from (F', \mathcal{R}) such that $\text{trig}(\mathcal{D}') = \mathcal{T}$. By enforcing the transitory and the active factbases of every element of \mathcal{D}' to be equal (which is trivially possible, see Remark 4.1) we produce an **O**-derivation \mathcal{D}' from (F', \mathcal{R}) . This oblivious chase derivation is uniquely defined by \mathcal{T} , hence we can name it the *oblivious restriction* of \mathcal{D} induced by F' .

Definition 5.5. Let \mathcal{D} be a derivation from (F, \mathcal{R}) and $F' \subseteq F$. The **O**-derivation \mathcal{D}'_F whose sequence of associated triggers from (F', \mathcal{R}) is equal to the restriction of $\text{trig}(\mathcal{D})$ induced by F' is called the *oblivious restriction* of \mathcal{D} induced by F' . \dashv

In fact the **O**-derivations are those that allow for more triggers to be applied (as they do not remove any atoms from the active factbase and they do not impose any extra condition of applicability). Therefore when searching for a *maximal* subsequence of $\text{trig}(\mathcal{D})$ to be applicable on a particular knowledge base, we can without loss of generality search only among **O**-derivations. And in **O**-derivations, changing the order of rule applications (when possible with respect to the ancestor/descendant relations), does not influence the (**O**-)applicability of the rest of the triggers. In other words, unlike in almost all other chase variants, in the oblivious chase, the prioritization of some triggers does not render other triggers non-applicable. From this we can conclude

that there is always a unique maximal subsequence, i.e. a unique restriction of $\text{trig}(\mathcal{D})$ induced by a particular F' .

Definition 5.6 (Hereditry). A monotonic chase variant X is said to be *hereditary* if, for any X -derivation \mathcal{D} from (F, \mathcal{R}) , for every subset $F' \subseteq F$, the oblivious restriction of \mathcal{D} induced by F' is an X -derivation. \dashv

So a chase is hereditary if by restricting an X -derivation on a subset of a factbase we still obtain an X -derivation. This property is fulfilled by the oblivious, the semi-oblivious and the restricted chase variant.

Proposition 5.3. The X -chase is hereditary for $X \in \{\mathbf{O}, \mathbf{bf-O}, \mathbf{SO}, \mathbf{R}\}$.

Proof: We assume that \mathcal{D} is an X -derivation from F and \mathcal{R} , and \mathcal{T} is the restriction of $\text{trig}(\mathcal{D})$ induced by $F' \subseteq F$. Let \mathcal{D}' be the oblivious restriction of \mathcal{D} induced by F' .

Case $X=\mathbf{O}$: Clearly \mathcal{D}' is an \mathbf{O} -chase derivation, therefore the \mathbf{O} -chase is hereditary.

Case $X=\mathbf{bf-O}$: Since \mathcal{D} is rank compatible and since the ordering of triggers is preserved in \mathcal{D}' , we get that \mathcal{D}' is rank compatible. Moreover, because \mathcal{D} is a $\mathbf{bf-O}$ -derivation, all triggers which are descendants of F' with rank at most the depth of \mathcal{D} do appear in \mathcal{D} . Therefore \mathcal{D}' is also breadth-first, since at every rank, all possible rule applications (up to its depth) from F' are made.

Case $X=\mathbf{SO}$: The condition for \mathbf{SO} -applicability is that we do not have two triggers from the same rule mapping frontier variables in the same way. We know that $\text{trig}(\mathcal{D})$ fulfills this condition, hence so does its subsequence $\text{trig}(\mathcal{D}')$.

Case $X=\mathbf{R}$: The condition for \mathbf{R} -applicability imposes that for a trigger $\mathbf{t} = (R, \pi)$ there is no extension of π that maps the head of R to F . Since \mathcal{D}' generates a factbase included in the factbase generated by \mathcal{D} we conclude that \mathbf{R} -applicability is preserved. \square

To conclude this paragraph, we show that hereditry implies preservation of ancestry:

Proposition 5.4. Every hereditary chase variant preserves ancestry.

Proof: Let X be a hereditary chase variant. Let \mathcal{D} be an X -derivation from (F, \mathcal{R}) and \mathfrak{t} be a trigger that produces atom A in \mathcal{D} . Let $F' = \text{Anc}_{\mathcal{D}}^0(A)$. Since X is hereditary, there exists an X -derivation \mathcal{D}' from (F', \mathcal{R}) with $\text{trig}(\mathcal{D}')$ being the restriction of $\text{trig}(\mathcal{D})$ induced by F' . Let \mathcal{T}_A be the subsequence of $\text{trig}(\mathcal{D})$ which contains all the triggers that produced any ancestor of A in \mathcal{D} as well as the trigger that produced A in \mathcal{D} . From Lemma 4.2 we know that \mathcal{T}_A is applicable on (F', \mathcal{R}) , therefore it is a subsequence of $\text{trig}(\mathcal{D}')$ which is the maximal subsequence of $\text{trig}(\mathcal{D})$ to be applicable on (F', \mathcal{R}) . Therefore $\mathfrak{t} \in \text{trig}(\mathcal{D}')$. Finally we must show that $\text{rank}_{\mathcal{D}}(A) = \text{rank}_{\mathcal{D}'}(A)$. At first notice that a consequence of the fact that $\text{trig}(\mathcal{D}')$ is a subsequence of $\text{trig}(\mathcal{D})$ is that since \mathfrak{t} produces A in \mathcal{D} , it must also produce A in \mathcal{D}' (otherwise there would be a trigger \mathfrak{t}' appearing before \mathfrak{t} in $\text{trig}(\mathcal{D}')$ and producing A , but this same trigger would appear before \mathfrak{t} also in $\text{trig}(\mathcal{D})$, so A would not be produced by \mathfrak{t} in \mathcal{D} which is a contradiction). We will prove an even more general statement:

Statement I: For every trigger $\mathfrak{t} \in \text{trig}(\mathcal{D}')$, the set $\mathbb{A}_{\mathfrak{t}}$ of all the atoms produced by \mathfrak{t} in \mathcal{D} is a subset of the set $\mathbb{A}'_{\mathfrak{t}}$ of all the atoms that are produced by \mathfrak{t} in \mathcal{D}' .

Let $\text{trig}(\mathcal{D}) = \mathfrak{t}_1, \mathfrak{t}_2, \dots, \mathfrak{t}_m$ and $\text{trig}(\mathcal{D}') = \mathfrak{t}'_1, \mathfrak{t}'_2, \dots, \mathfrak{t}'_n$. Let $0 \leq i \leq n$ and suppose that $\mathfrak{t}'_i = \mathfrak{t}_{\ell}$. Then it holds that

$$\mathbb{A}'_{\mathfrak{t}'_i} = \text{op}(\mathfrak{t}'_i) \setminus (F' \cup \text{op}(\mathfrak{t}'_1) \cup \text{op}(\mathfrak{t}'_2) \cup \dots \cup \text{op}(\mathfrak{t}'_{i-1}))$$

whereas

$$\mathbb{A}_{\mathfrak{t}'_i} = \text{op}(\mathfrak{t}'_i) \setminus (F \cup \text{op}(\mathfrak{t}_1) \cup \text{op}(\mathfrak{t}_2) \cup \dots \cup \text{op}(\mathfrak{t}_{\ell-1}))$$

but $\mathfrak{t}'_1, \mathfrak{t}'_2, \dots, \mathfrak{t}'_{i-1}$ is a subsequence of $\mathfrak{t}_1, \mathfrak{t}_2, \dots, \mathfrak{t}_{\ell-1}$ so

$$F' \cup \text{op}(\mathfrak{t}'_1) \cup \dots \cup \text{op}(\mathfrak{t}'_{i-1}) \subseteq F \cup \text{op}(\mathfrak{t}_1) \cup \dots \cup \text{op}(\mathfrak{t}_{\ell-1})$$

therefore $\mathbb{A}_{t'_i} \subseteq \mathbb{A}'_{t'_i}$ for every $i \leq n$, hence we have proved that $\mathbb{A}_t \subseteq \mathbb{A}'_t$ for every $t \in \text{trig}(\mathcal{D}')$. We can use this result to prove that

Statement II: For every pair A_1, A_2 of ancestors of A in \mathcal{D} , if A_1 is a direct ancestor of A_2 in \mathcal{D} , then A_1 is also a direct ancestor of A_2 in \mathcal{D}' .

The argumentation to show this statement is simpler: let t be the trigger that produces A_2 in \mathcal{D} . We have that $t \in \mathcal{T}_A$ so $t \in \text{trig}(\mathcal{D}')$. Hence, in order to show that A_1 is a direct ancestor of A_2 in \mathcal{D}' , it suffices to show that t produces A_2 in \mathcal{D}' . This holds because $\mathbb{A}_t \subseteq \mathbb{A}'_t$ and $A_2 \in \mathbb{A}_t$.

Let G be the chase graph associated with \mathcal{D} and G' the chase graph associated with \mathcal{D}' . The second statement implies that the subgraphs of G and G' that are induced by the ancestors of A in \mathcal{D} and \mathcal{D}' respectively are equal. This guarantees that the rank of A in both derivations is the same. We have shown that there exists a derivation \mathcal{D}' from $(\text{Anc}_{\mathcal{D}}^0(A), \mathcal{R})$ that produces A in the same rank as \mathcal{D} , hence we conclude that X preserves ancestry. \square

Corollary 5.1. X - k -Boundedness is decidable when $X \in \{\text{O}, \text{bf-O}, \text{SO}, \text{R}\}$.♣

5.2.3 bf-R-Compliance

We proved that both oblivious chase and breadth-first oblivious chase are hereditary. Does the breadth-first approach affect the property of heredity in the semi-oblivious and the restricted chase? The answer is yes. Indeed, given some hereditary chase variant X , if \mathcal{D} is a breadth-first X -derivation, then the restriction of $\text{trig}(\mathcal{D})$ induced by F' does not necessarily produce a breadth-first X -derivation, as shown by the next examples, 34 and 35. The oblivious chase is the only exception among the chase variants that we study.

Example 34: bf-SO-chase. Let $F = \{p(a, b), r(a, c)\}$ and \mathcal{R} is:

$$R_1 = p(x, y) \rightarrow r(x, y)$$

$$R_2 = r(x, y) \rightarrow \exists z q(x, z)$$

$$R_3 = r(x, y) \rightarrow t(y)$$

Below is the **bf-SO**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{r(a, b)\}$	$Z_1 = F_1$	
$\mathfrak{t}_2 = (R_3, \{x \mapsto a, y \mapsto c\})$	$F_2 = F_1 \cup \{t(c)\}$	$Z_2 = F_2$	
$\mathfrak{t}_3 = (R_2, \{x \mapsto a, y \mapsto c\})$	$F_3 = F_2 \cup \{q(c, z_{\mathfrak{t}_3})\}$	$Z_3 = F_3$	1
$\mathfrak{t}_4 = (R_3, \{x \mapsto a, y \mapsto b\})$	$F_4 = F_3 \cup \{t(b)\}$	$Z_4 = F_4$	2

\mathcal{D} is a terminating **bf-SO**-derivation of depth 2. Let $F' = \{p(a, b)\}$. The restriction of \mathcal{D} induced by F' includes only \mathfrak{t}_1 and \mathfrak{t}_4 . Applying those triggers in a monotonic fashion we obtain \mathcal{D}' :

\emptyset	$F'_0 = F'$	$Z'_0 = F'_0$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F'_1 = F' \cup \{r(a, b)\}$	$Z'_1 = F'_1$	1
$\mathfrak{t}_4 = (R_3, \{x \mapsto a, y \mapsto b\})$	$F'_2 = F'_1 \cup \{t(b)\}$	$Z'_2 = F'_2$	2

\mathcal{D}' is a rank compatible **SO**-derivation of depth 2 from (F', \mathcal{R}) , however it is not breadth-first since now $\mathfrak{t}_5 = (R_2, \{x \mapsto a, y \mapsto b\})$ is **SO**-applicable at rank 2 (from Lemma 4.1). ■

Example 35: bf-R-chase. Let $F = \{p(a, b), q(a, c)\}$ and \mathcal{R} is:

$$R_1 = p(x, y) \rightarrow r(x, y)$$

$$R_2 = r(x, y) \rightarrow \exists z q(x, z)$$

$$R_3 = r(x, y) \rightarrow t(x)$$

Let $\pi = \{x \mapsto a, y \mapsto b\}$. Below is the **bf-R**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathfrak{t}_1 = (R_1, \pi)$	$F_1 = F \cup \{r(a, b)\}$	$Z_1 = F_1$	1
$\mathfrak{t}_2 = (R_3, \pi)$	$F_2 = F_1 \cup \{t(a)\}$	$Z_2 = F_2$	2

\mathcal{D} is a terminating **bf-R**-derivation of depth 2. Notice that the trigger $\mathfrak{t}_3 = (R_2, \pi)$ is **SO**-applicable on \mathcal{D} , but not **R**-applicable because of the presence of $q(a, c)$ in $Z^\mathcal{D}$.

Let $F' = \{p(a, b)\}$. In this case the restriction of $\text{trig}(\mathcal{D})$ induced by F' is equal to itself! Hence we obtain the similar derivation \mathcal{D}' from (F, \mathcal{R}) :

\emptyset	$F'_0 = F'$	$Z'_0 = F'_0$	0
$\mathbf{t}_1 = (R_1, \pi)$	$F'_1 = F' \cup \{r(a, b)\}$	$Z'_1 = F'_1$	1
$\mathbf{t}_2 = (R_3, \pi)$	$F'_2 = F'_1 \cup \{t(a)\}$	$Z'_2 = F'_2$	2

Again \mathcal{D}' is a **R**-derivation of depth 2, however it is not breadth-first since now \mathbf{t}_3 is **R**-applicable at rank 2 and thus has to be applied to ensure that we have exhausted all applications of the final rank (according to Lemma 4.1). ■

As we have seen, heredity is a sufficient property to ensure decidability of k -boundedness. However it is not general enough, as it does not account for breadth-first derivations or any other kind of prioritization of certain triggers over others.

On the other hand, several breadth-first chase variants have features that assure preservation of ancestry. In what follows, we employ the chase space perspective and pertinent results from Section 4.5. At first we define a class of chase variants, which comprises various useful features. We introduce **bf-R-compliance** as a property on chase variants that guarantees that every derivation includes at least the triggers that appear in a **bf-R**-derivation from the same knowledge base. More formally:

Definition 5.7 (**bf-R-compliance**). The X -chase is **bf-R-compliant** if

- it is a breadth-first submonotonic chase variant,
- for every finite X -derivation \mathcal{D} , if a trigger $\mathbf{t} \notin \text{trig}(\mathcal{D})$ is **R**-applicable on $Z^{\mathcal{D}}$, then \mathbf{t} is also X -applicable on \mathcal{D} . \dashv

Note that the last element of a breadth-first derivation is necessarily a rank mark, so the second of the conditions above implies that for every rank mark of an X -derivation \mathcal{D} , every trigger that is **R**-applicable on the active factbase at that point, is also X -applicable on the respective prefix of \mathcal{D} . Oftentimes, we will use this statement in contraposition, i.e. if a trigger is not X -applicable at this point, then neither is it **R**-applicable on the active factbase. This means that all the triggers of the current rank that are not applied in the derivation, have the property that their output can be folded back to the active factbase with a

retraction. Because this has to be true for all ranks, we are effectively forcing the derivation to include at least all the triggers that a **bf-R**-derivation includes.

Directly from their definitions, we can verify that the following chase variants are **bf-R**-compliant:

Remark 5.2. If $X \in \{\mathbf{bf-O}, \mathbf{bf-SO}, \mathbf{bf-R}, \mathbf{P}\}$, then the X -chase is **bf-R**-compliant. ♣

We will show that k -boundedness is decidable for all these chase variants by utilizing again preservation of ancestry (of course the case of **bf-O**-chase has already been shown with heredity).

Now we will prove several nice properties that stem from **bf-R**-compliance. In particular the following four properties (i.e. Theorem 5.3, Corollary 5.2, Theorem 5.4 and Proposition 5.5) are instrumental to proving the preservation

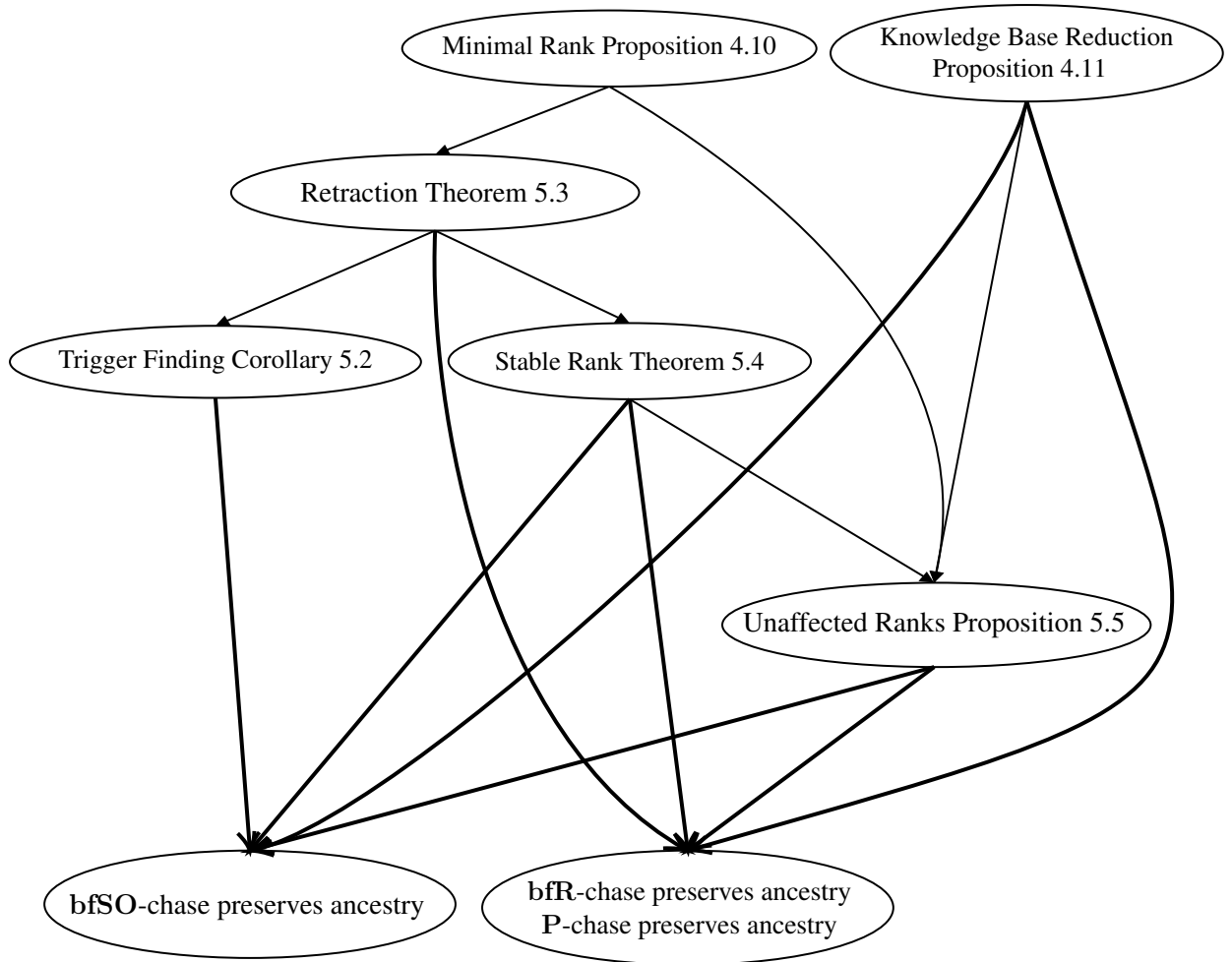


Figure 5.1: Properties used to show preservation of ancestry of **bf-R**-compliant chase variants.

of ancestry for the **bf-SO**-, the **bf-R**- and the **P**-chase at the end of this section. Two more propositions from the previous chapter are also employed and in Figure 5.1 we illustrate the interdependency of all those properties.

Theorem 5.3 (The Retraction Theorem). Let X be a chase variant that is **bf-R**-compliant. Let \mathcal{D} be an X -derivation from a knowledge base (F, \mathcal{R}) . Let \mathcal{D}' be a derivation from (F, \mathcal{R}) with depth less than or equal to the depth of \mathcal{D} . Then there exists a retraction h from $Z^{\mathcal{D}'} \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$ such that for every $A \in Z^{\mathcal{D}'}$ holds that $\text{rank}_{\mathcal{D}}(h(A)) \leq \text{rank}_{\mathcal{D}'}(A)$.

Proof: In this proof, for any (rank) j , we use the notation \mathcal{D}^j to denote the prefix of derivation \mathcal{D} which includes all elements of rank at most j .

It suffices to show that the above theorem holds when \mathcal{D}' is a **bf-O**-derivation of depth equal to \mathcal{D} . That is because the (final) active factbase of any other derivation of equal or smaller depth from (F, \mathcal{R}) is a subset of the final factbase of \mathcal{D}' with greater or equal ranks for the common atoms (by Proposition 4.10).

Let $\mathcal{T} = \text{trig}(\mathcal{D}') \setminus \text{trig}(\mathcal{D})$.

We do induction on the depth k of \mathcal{D} (and \mathcal{D}').

- [Base case: $k = 1$] Every trigger $\mathfrak{t}_\nu \in \mathcal{T}$ has to be applicable on F , thus, we have that $\text{sp}(\mathfrak{t}_\nu) \subseteq F$. We know that \mathfrak{t}_ν is not X -applicable on \mathcal{D} , because it is not in $\text{trig}(\mathcal{D})$ and \mathcal{D} is breadth-first with depth at least 1. Therefore, since X is a chase variant that is **bf-R**-compliant, we have that \mathfrak{t}_ν is not **R**-applicable on $Z^{\mathcal{D}}$. But it is applicable on $Z^{\mathcal{D}}$, because from the submonotonicity of the X -chase we know that $F \subseteq Z^{\mathcal{D}}$. Hence there is a retraction $h_\nu : \text{op}(\mathfrak{t}_\nu) \cup Z^{\mathcal{D}} \rightarrow Z^{\mathcal{D}}$. As a result for every $A \in \text{op}(\mathfrak{t}_\nu)$ we also have that $\text{rank}_{\mathcal{D}}(h_\nu(A)) \leq 1$. But for every such A that is produced by \mathfrak{t}_ν in \mathcal{D}' , we know that $\text{rank}_{\mathcal{D}'}(A) = 1$. This confirms that for every such A holds $\text{rank}_{\mathcal{D}'}(A) \geq \text{rank}_{\mathcal{D}}(h_\nu(A))$. Finally, because if $\mathfrak{t}_\nu \neq \mathfrak{t}'_\nu$, then $\text{nul}(\text{op}(\mathfrak{t}_\nu))$ is disjoint with $\text{nul}(\text{op}(\mathfrak{t}'_\nu))$, we can compose all such retractions h_ν for every $\mathfrak{t}_\nu \in \mathcal{T}$ producing the required h .
- [Induction step: we suppose the property holds for some $k - 1$] We suppose that \mathcal{D} and \mathcal{D}' are of depth k . Since the property holds for

$k - 1$, we know that there exists a retraction $h : Z^{\mathcal{D}'^{k-1}} \cup Z^{\mathcal{D}^{k-1}} \rightarrow Z^{\mathcal{D}^{k-1}}$ with the property that for every atom $A \in Z^{\mathcal{D}'^{k-1}}$ holds that $\text{rank}_{\mathcal{D}'}(A) \geq \text{rank}_{\mathcal{D}}(h(A))$.

Let $\bar{\mathcal{T}} = \{\mathfrak{t} \in \text{trig}(\mathcal{D}') \setminus \text{trig}(\mathcal{D}) \mid \mathfrak{t} \notin \text{trig}(\mathcal{D}'^{k-1})\}$. Let $\mathfrak{t}_\nu \in \bar{\mathcal{T}}$ and $\mathfrak{t}_\nu = (R_\nu, \pi_\nu)$. Since \mathfrak{t}_ν is of rank k in \mathcal{D}' , \mathfrak{t}_ν is applicable on $Z^{\mathcal{D}'^{k-1}}$, so $\text{sp}(\mathfrak{t}_\nu) \subseteq Z^{\mathcal{D}'^{k-1}}$. Therefore $h(\text{sp}(\mathfrak{t}_\nu)) \subseteq Z^{\mathcal{D}^{k-1}}$. As a result the trigger $\mathfrak{t}'_\nu = (R_\nu, h \circ \pi_\nu)$, is applicable on $Z^{\mathcal{D}^{k-1}}$. Notice that $\text{op}(\mathfrak{t}'_\nu) = s_\nu(h(\text{op}(\mathfrak{t}_\nu)))$, where $s_\nu : \text{nul}(\text{op}(\mathfrak{t}_\nu)) \rightarrow \text{nul}(\text{op}(\mathfrak{t}'_\nu))$ is a simple renaming with $s(x_{\mathfrak{t}_\nu}) = x_{\mathfrak{t}'_\nu}$ for every $x_{\mathfrak{t}_\nu} \in \text{nul}(\text{op}(\mathfrak{t}_\nu))$. Again here we have two cases:

- i) $\mathfrak{t}'_\nu \in \text{trig}(\mathcal{D})$. We assume that \mathfrak{t}'_ν is of rank ℓ in \mathcal{D} (so $\ell \leq k$). In this case $\text{op}(\mathfrak{t}'_\nu) \in F^{\mathcal{D}^\ell}$. By definition (see Definition 4.1), there is a retraction σ from $F^{\mathcal{D}^\ell}$ to $Z^{\mathcal{D}^\ell}$, so also $\sigma(\text{op}(\mathfrak{t}'_\nu)) \subseteq Z^{\mathcal{D}^\ell}$. Let σ_ν be the restriction of σ to $\text{var}(\text{op}(\mathfrak{t}'_\nu))$. The domain of σ_ν is disjoint with $\text{var}(Z^{\mathcal{D}^{\ell-1}})$ because X is submonotonic. Therefore σ_ν only affects variables of $\text{nul}(\text{op}(\mathfrak{t}_\nu))$, so $\sigma_\nu \circ s_\nu \circ h$ is a retraction from $\text{op}(\mathfrak{t}_\nu) \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$.
- ii) $\mathfrak{t}'_\nu \notin \text{trig}(\mathcal{D})$. Because \mathcal{D} is breadth-first, we get that \mathfrak{t}'_ν is not X -applicable on \mathcal{D} . So it is not \mathbf{R} -applicable on $Z^{\mathcal{D}}$, since X is $\mathbf{bf-R}$ -compliant. Moreover, the submonotonicity of the X -chase guarantees that \mathfrak{t}'_ν is applicable on $Z^{\mathcal{D}}$. So there is a retraction $\sigma_\nu : \text{op}(\mathfrak{t}'_\nu) \cup Z^{\mathcal{D}} \rightarrow Z^{\mathcal{D}}$. Therefore $\sigma_\nu \circ s_\nu \circ h$ is a retraction from $\text{op}(\mathfrak{t}_\nu) \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$.

We see that in both cases, there exists a retraction $h_\nu = \sigma_\nu \circ s_\nu$ from $\text{op}(\mathfrak{t}_\nu) \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$. So $\text{rank}_{\mathcal{D}}(h(A')) \leq k$ for every $A' \in \text{op}(\mathfrak{t}_\nu)$. If A' is produced by \mathfrak{t}_ν in \mathcal{D}' we have $\text{rank}_{\mathcal{D}'}(A') = k$. This assures that $\text{rank}_{\mathcal{D}'}(A') \geq \text{rank}_{\mathcal{D}}(h(A'))$. Moreover, the domains of all h_ν (corresponding to each different $\mathfrak{t}_\nu \in \bar{\mathcal{T}}$) are pairwise disjoint and (because they are retractions) they are all disjoint with their codomain, which implies that they can be composed in parallel, so if $\bar{\mathcal{T}} = \{\mathfrak{t}_{\nu_1}, \mathfrak{t}_{\nu_2}, \dots, \mathfrak{t}_{\nu_\omega}\}$,

then $\bar{h} = h_{\nu_1} \circ h_{\nu_2} \circ \dots \circ h_{\nu_\omega}$ is a well defined substitution whose domain is in $\mathbf{nul}(Z^{\mathcal{D}'}) \setminus \mathbf{nul}(Z^{\mathcal{D}'^{k-1}})$. So the composition $h' = \bar{h} \circ h$ is also a well defined substitution from $\mathbf{nul}(Z^{\mathcal{D}'}) \setminus \mathbf{nul}(Z^{\mathcal{D}})$ to $\mathbf{term}(Z^{\mathcal{D}})$. Finally, we have that $h'(Z^{\mathcal{D}'}) \subseteq Z^{\mathcal{D}}$, so h' is a retraction from $Z^{\mathcal{D}'} \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$ and it also holds that $\text{rank}_{\mathcal{D}'}(A') \geq \text{rank}_{\mathcal{D}}(h(A'))$ for every A' produced by a $\mathbf{t}_\nu \in \mathcal{T}$ in \mathcal{D}' .

The induction is complete and so is the proof. \square

With the following corollary we extend the previous result, showing that every trigger of the chase space that does not appear in an X-derivation \mathcal{D} (where X is **bf-R**-compliant) corresponds to a trigger that is applicable during \mathcal{D} , i.e. whose support is inferred by \mathcal{D} .

Corollary 5.2 (Trigger Finding). Let X be a chase variant that is **bf-R**-compliant. Let \mathcal{D} be an X-derivation on (F, \mathcal{R}) of depth at least m_0 . Then for every $\mathbf{t} = (R, \pi) \in \mathcal{R}^F \setminus \mathbf{trig}(\mathcal{D})$ with $\text{rank}_{(F, \mathcal{R})}(\mathbf{t}) \leq m_0$, there exists a retraction h from $\mathbf{sp}(\mathbf{t}) \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$ such that $\mathbf{t}' = (R, h \circ \pi) \in \mathcal{R}^F$ with $\text{rank}_{(F, \mathcal{R})}(\mathbf{t}') \leq \text{rank}_{(F, \mathcal{R})}(\mathbf{t})$.

Proof: Since the ranks of triggers and atoms in a chase space are those of a breadth-first oblivious derivation, this proof involves comparing the X-derivation \mathcal{D} from (F, \mathcal{R}) with an exhaustive **bf-O**-derivation \mathcal{D}' from the same knowledge base.

The derivation \mathcal{D}' must include \mathbf{t} , because it is an exhaustive **O**-derivation. Let $\text{rank}_{\mathcal{D}'}(\mathbf{t}) = \text{rank}_{(F, \mathcal{R})}(\mathbf{t}) = k + 1$ (so $k < m_0$). We use the notation \mathcal{D}^k to represent the prefix of \mathcal{D} including all elements of rank at most k . Respectively, \mathcal{D}'^k to represent the prefix of \mathcal{D}' including all elements of rank at most k . Since \mathcal{D}^k is an X-derivation and \mathcal{D}'^k is of the same depth, we can apply Theorem 5.3 and conclude that there exists a retraction h from $Z^{\mathcal{D}^k} \cup Z^{\mathcal{D}'^k}$ to $Z^{\mathcal{D}'^k}$ such that for every $A \in Z^{\mathcal{D}^k}$ it holds that

$$\text{rank}_{\mathcal{D}^k}(h(A)) \leq \text{rank}_{\mathcal{D}'^k}(A) \quad (5.1)$$

We have that $\mathbf{sp}(\mathbf{t}) \subseteq Z^{\mathcal{D}'^k}$ so h is also a retraction from $\mathbf{sp}(\mathbf{t}) \cup Z^{\mathcal{D}^k}$ to $Z^{\mathcal{D}'^k}$.

Moreover let $\mathfrak{t}' = (R, h \circ \pi)$. We have that $\mathbf{sp}(\mathfrak{t}') = h(\mathbf{sp}(\mathfrak{t}))$ so \mathfrak{t}' is applicable on $Z^{\mathcal{D}}$, thus $\mathfrak{t}' \in \mathcal{R}^F$. Finally, from (5.1) we know that the maximum rank among atoms of $\mathbf{sp}(\mathfrak{t}')$ is bounded by the maximum rank of atoms in $\mathbf{sp}(\mathfrak{t})$, hence $\text{rank}_{(F, \mathcal{R})}(\mathfrak{t}') \leq \text{rank}_{(F, \mathcal{R})}(\mathfrak{t})$. \square

In the end of the previous chapter we discussed the ranks of triggers and atoms in a chase space and whether we could specify a class of derivations larger than **bf-O**-chase, which preserves these ranks. **bf-R**-compliance is convenient in this respect. The following theorem, establishes this important connection between all **X**-derivations, when **X** is **bf-R**-compliant. In Proposition 4.10 we saw that the ranks of atoms in the chase space are the minimal ranks that can be achieved by any derivation from this knowledge base. Below we show that all derivations of **bf-R**-compliant chase variants necessarily produce atoms at that minimal rank.

Theorem 5.4 (Stable Rank Theorem). Let **X** be a chase variant that is **bf-R**-compliant. Let $\mathbb{C}(F, \mathcal{R})$ be a chase space and let \mathcal{D} be an **X**-derivation on (F, \mathcal{R}) . Then for every $A \in F^{\mathcal{D}}$ holds that $\text{rank}_{\mathcal{D}}(A) = \text{rank}_{(F, \mathcal{R})}(A)$.

Proof: In what follows we will use the notation \mathcal{D}^i to represent the prefix of \mathcal{D} including all elements of rank up to i . Note that for every atom $A_0 \in F^{\mathcal{D}^i}$ holds that $\text{rank}_{\mathcal{D}^i}(A_0) = \text{rank}_{\mathcal{D}}(A_0)$.

Since the ranks of triggers and atoms in a chase space are those of a breadth-first oblivious derivation, this proof involves comparing ranks in the **bf-X**-derivation \mathcal{D} from (F, \mathcal{R}) with ranks in a **bf-O**-derivation from the same knowledge base.

We know that the first trigger \mathfrak{t}_1 of $\mathbf{trig}(\mathcal{D})$ that produces an atom, is of rank 1, as all of $\mathbf{sp}(\mathfrak{t}_1)$ is in the initial factbase F . So then all of the atoms produced by \mathfrak{t}_1 are necessarily of rank 1 in $\mathbb{C}(F, \mathcal{R})$, as they do not belong to F but can be produced by the application of one rule on F .

Let \mathfrak{t} be the first trigger in $\mathbf{trig}(\mathcal{D})$ that produces an atom A such that $\text{rank}_{\mathcal{D}}(A) \neq \text{rank}_{(F, \mathcal{R})}(A)$. But every atom $A' \in \mathbf{sp}(\mathfrak{t})$ is produced earlier in \mathcal{D} , so $\text{rank}_{\mathcal{D}}(A') = \text{rank}_{(F, \mathcal{R})}(A')$, and so there is an exhaustive **bf-O**-derivation \mathcal{D}' that produces all $\mathbf{sp}(\mathfrak{t})$ in the same ranks as \mathcal{D} ,

as a result \mathfrak{t} is applicable on a prefix of \mathcal{D}' , hence it appears in \mathcal{D}' and $\text{rank}_{\mathcal{D}'}(\mathfrak{t}) = \text{rank}_{\mathcal{D}}(\mathfrak{t})$. We want to show that $\text{rank}_{\mathcal{D}'}(A) = \text{rank}_{\mathcal{D}}(A)$. If \mathfrak{t} indeed produces A in \mathcal{D}' , then the result follows. Otherwise if \mathfrak{t} does not produce A in \mathcal{D}' , then there is a trigger \mathfrak{t}' , which appears before \mathfrak{t} in $\text{trig}(\mathcal{D}')$ and produces A . We split to two cases:

- If $\text{rank}_{\mathcal{D}'}(\mathfrak{t}') = \text{rank}_{\mathcal{D}'}(\mathfrak{t})$, then because $\text{rank}_{\mathcal{D}'}(A) = \text{rank}_{\mathcal{D}'}(\mathfrak{t}')$ and $\text{rank}_{\mathcal{D}}(\mathfrak{t}) = \text{rank}_{\mathcal{D}}(A)$, we conclude that $\text{rank}_{\mathcal{D}'}(A) = \text{rank}_{\mathcal{D}}(A)$.
- For the case of $\text{rank}_{\mathcal{D}'}(\mathfrak{t}') < \text{rank}_{\mathcal{D}'}(\mathfrak{t})$, we will arrive at contradiction.

At first, we know that A does not include any (new) variables indexed by \mathfrak{t} or \mathfrak{t}' , since it appears in the output of both triggers. Let $\text{rank}_{\mathcal{D}}(\mathfrak{t}) = k + 1$. Then all the variables in A are already created (and present) in $Z^{\mathcal{D}^k}$. Furthermore, Proposition 4.10 implies that $\text{rank}_{\mathcal{D}'}(\mathfrak{t}) \leq k + 1$, hence $\text{rank}_{\mathcal{D}'}(\mathfrak{t}') \leq k$. So $A \in Z^{\mathcal{D}'^k}$.

From Theorem 5.3 we know that there exists a retraction h from $Z^{\mathcal{D}'^k} \cup Z^{\mathcal{D}^k}$ to $Z^{\mathcal{D}^k}$. Because the variables of A appear in $Z^{\mathcal{D}^k}$, we know that $h(A) = A$. But then $A \in Z^{\mathcal{D}^k}$ which contradicts the fact that A is produced by \mathfrak{t} in \mathcal{D} . As a result we cannot have a trigger of rank lower than that of \mathfrak{t} , producing A in \mathcal{D}' , so it has to be that $\text{rank}_{\mathcal{D}'}(A) = \text{rank}_{\mathcal{D}'}(\mathfrak{t})$ therefore $\text{rank}_{\mathcal{D}'}(A) = \text{rank}_{\mathcal{D}}(A)$. The proof is complete. \square

In Example 30, we saw that the same triggers have different ranks in different chase spaces. In order to prove the preservation of ancestry for three bf-R-compliant chase variants, we first provide a proposition which guarantees the preservation of the ranks of the triggers producing the ancestors of an atom, when we switch from the original chase space to a particular smaller one.

Proposition 5.5 (Unaffected Ranks of Ancestors). Let X be a chase variant that is bf-R-compliant. Let \mathcal{D} be an X -derivation, $A \in F^{\mathcal{D}}$ and $F' = \text{Anc}_{\mathcal{D}}^0(A)$. Let \mathcal{T}_A be the subsequence of $\text{trig}(\mathcal{D})$ that contains all the triggers that produced any ancestor of A in \mathcal{D} as well as the trigger that produced A in \mathcal{D} . Then for every trigger $\mathfrak{t} \in \mathcal{T}_A$ holds that $\text{rank}_{\mathcal{D}}(\mathfrak{t}) = \text{rank}_{(F', \mathcal{R})}(\mathfrak{t})$ and for every $A' \in \text{Anc}_{\mathcal{D}}(A)$ holds that $\text{rank}_{\mathcal{D}}(A') = \text{rank}_{(F', \mathcal{R})}(A')$.

Proof: Let \mathcal{D}_A be the **O**-derivation from (F', \mathcal{R}) such that $\text{trig}(\mathcal{D}_A) = \mathcal{T}_A$. At first we will prove by induction that for every $\mathfrak{t} \in \mathcal{T}_A$ and $A' \in \text{op}(\mathfrak{t}) \cap \text{Anc}_{\mathcal{D}}(A)$ holds that $\text{rank}_{\mathcal{D}}(\mathfrak{t}) = \text{rank}_{\mathcal{D}_A}(\mathfrak{t})$ and $\text{rank}_{\mathcal{D}}(A') = \text{rank}_{\mathcal{D}_A}(A')$.

Let $\mathcal{T}_A = \mathfrak{t}_1, \mathfrak{t}_2, \dots, \mathfrak{t}_n$. Since \mathfrak{t}_1 is the first trigger in $\text{trig}(\mathcal{D})$ that produces an ancestor of A in \mathcal{D} , we know that its support is included in $\text{Anc}_{\mathcal{D}}^0(A)$ and it is of rank 1 in \mathcal{D} . Thus $\text{sp}(\mathfrak{t}) \in F'$ so in both derivations the rank of \mathfrak{t} is equal to 1. Moreover let $A' \in \text{op}(\mathfrak{t}) \cap \text{Anc}_{\mathcal{D}}(A)$. If A' is in $\text{Anc}_{\mathcal{D}}^0(A)$ then it is of rank 0 in both derivations whereas if $A' \notin \text{Anc}_{\mathcal{D}}^0(A)$ then it has to be of rank 1 in \mathcal{D}' (because it is produced by \mathfrak{t}) and also of rank 1 in \mathcal{D} (because \mathfrak{t} is the first trigger in \mathcal{D} that produces an ancestor of A , so A' cannot have been produced before).

Suppose now that the equalities are true for triggers $\mathfrak{t}_1, \mathfrak{t}_2, \dots, \mathfrak{t}_{i-1}$ and all atoms from $\text{Anc}_{\mathcal{D}}(A)$ that they produce. Because \mathcal{T}_A includes all the triggers that produce ancestors of A in \mathcal{D} , we have that

$$\text{sp}(\mathfrak{t}_i) \in (F' \cup \text{op}(\mathfrak{t}_1) \cup \text{op}(\mathfrak{t}_2) \cup \dots \cup \text{op}(\mathfrak{t}_{i-1})) \cap \text{Anc}_{\mathcal{D}}(A)$$

therefore the ranks of the atoms in the support of \mathfrak{t}_i are equal in the two derivations, so it also holds that $\text{rank}_{\mathcal{D}}(\mathfrak{t}_i) = \text{rank}_{\mathcal{D}_A}(\mathfrak{t}_i)$. Moreover every $A' \in \text{op}(\mathfrak{t}_i) \cap \text{Anc}_{\mathcal{D}}(A)$ is indeed produced by \mathfrak{t}_i in \mathcal{D}_A (because if it was produced by an earlier trigger \mathfrak{t}' of \mathcal{T}_A , it would also have been produced by \mathfrak{t}' in \mathcal{D}). Hence the induction is complete.

Let $\mathfrak{t} \in \mathcal{T}_A$. From Proposition 4.10 we know that the rank of a trigger \mathfrak{t} in the chase space $\mathbb{C}(F', \mathcal{R})$ is the minimal rank that any derivation from (F', \mathcal{R}) can achieve for \mathfrak{t} , so $\text{rank}_{(F', \mathcal{R})}(\mathfrak{t}) \leq \text{rank}_{\mathcal{D}_A}(\mathfrak{t})$. By replacing $\text{rank}_{\mathcal{D}_A}(\mathfrak{t})$ with $\text{rank}_{\mathcal{D}}(\mathfrak{t})$ this last equation turns to

$$\text{rank}_{(F', \mathcal{R})}(\mathfrak{t}) \leq \text{rank}_{\mathcal{D}}(\mathfrak{t}) \quad (5.2)$$

Now notice that since X is **bf-R**-compliant we have that $\text{rank}_{(F, \mathcal{R})}(\mathfrak{t}) = \text{rank}_{\mathcal{D}}(\mathfrak{t})$ (this is based on Theorem 5.4). Then, from Proposition 4.11, we have that $\text{rank}_{(F, \mathcal{R})}(\mathfrak{t}) \leq \text{rank}_{(F', \mathcal{R})}(\mathfrak{t})$ which gives us

that

$$\text{rank}_{\mathcal{D}}(\mathfrak{t}) \leq \text{rank}_{(F', \mathcal{R})}(\mathfrak{t}) \quad (5.3)$$

From (5.3) and (5.2) we conclude the desired $\text{rank}_{\mathcal{D}}(\mathfrak{t}) = \text{rank}_{(F', \mathcal{R})}(\mathfrak{t})$. \square

In the previous subsection we showed that every hereditary chase variant preserves ancestry. It seems possible that if a chase variant is **bf-R-compliant** and produces a universal model, then it preserves ancestry. Nevertheless, this time we proceed in a less general manner, by showing directly that the chase variants that concern us preserve ancestry. As exhibited in Figure 5.1, the theorems and propositions that we proved above are instrumental to this conclusion.

Starting from an X-derivation \mathcal{D} which produces an atom A at a certain rank m , we will show that it is possible to define an X-derivation \mathcal{D}' , whose initial factbase includes only the 0-rank ancestors of A in \mathcal{D} , and that also produces A at rank m . We specify an algorithm that performs this task. The “Unaffected Ranks Proposition” 5.5, combined with the “Stable Rank Theorem” 5.4 guarantee that the ranks of all the ancestors of A in \mathcal{D} that appear also in \mathcal{D}' are the same in the two X-derivations. Note that this only holds because X is **bfR-compliant**. Finally, in order to verify that all the triggers that produce ancestors of A in \mathcal{D} are X-applicable at their respective turn in \mathcal{D}' , we will employ the “Retraction Theorem” 5.3.

Proposition 5.6. The X-chase preserves ancestry when $X \in \{\mathbf{bf-SO}, \mathbf{bf-R}, \mathbf{P}\}$.

Proof: We assume that \mathcal{D} is an X-derivation from (F, \mathcal{R}) and \mathfrak{t} is a trigger that produces atom A in \mathcal{D} . Let $F' = \text{Anc}_{\mathcal{D}}^0(A)$ and $\mathcal{T}_A = \mathfrak{t}_{n_1}, \mathfrak{t}_{n_2}, \dots, \mathfrak{t}_{n_n}$ be the subsequence of $\text{trig}(\mathcal{D})$ that contains all the triggers that produced any ancestor of A in \mathcal{D} as well as \mathfrak{t} . So \mathfrak{t} is the n -th trigger of \mathcal{T}_A .

Let $\text{rank}_{\mathcal{D}}(A) = m$. The algorithm below can be called the “breadth-first completion” of \mathcal{T}_A on (F', \mathcal{R}) , because it constructs a derivation by completing the given sequence of triggers in a breadth-first manner. We will show that its result is an X-derivation \mathcal{D}' such that \mathcal{T}_A is a subsequence of $\text{trig}(\mathcal{D}')$. We call the algorithm 1.X, because it is parametrized by the particular chase variant

$X \in \{\mathbf{bf}\text{-SO}, \mathbf{bf}\text{-R}, \mathbf{P}\}.$

ALGORITHM 1.X: Input: $(F', \mathcal{R}), \mathcal{T}_A = \mathbf{t}_{n_1}, \mathbf{t}_{n_2}, \dots, \mathbf{t}_{n_n}.$

- 1) Set $\mathcal{D}' = (\emptyset, F', F'), Z'_0 = F',$
 $i = 1$ (where i is the current size of \mathcal{D}'),
 $j = 1$ (where j is the index of the next trigger of \mathcal{T}_A to be added to \mathcal{D}').
- 2) **for** (rank) $k = 1$ to m ,
 - I) **while** there are at least two different triggers from \mathcal{R} that are X -applicable on \mathcal{D}' ,
 - i) **if** the trigger \mathbf{t}_{n_j} of \mathcal{T}_A is X -applicable on \mathcal{D}' ,
 - a) Set $\mathbf{t}'_i = \mathbf{t}_{n_j}, F'_i = Z'_{i-1} \cup \mathbf{op}(\mathbf{t}_{n_j}).$
 - b) $j++.$
 - c) **X-dependent step:** \cdot if $X \in \{\mathbf{bf}\text{-R}, \mathbf{bf}\text{-SO}\}$, set $Z_i = F_i.$
 \cdot if $X = \mathbf{P}$, set $Z_i = Z_{i-1}.$
 - d) Add $(\mathbf{t}'_i, F'_i, Z'_i)$ to $\mathcal{D}'.$
 - e) $i++.$
 - ii) **else if** there exists a trigger $\mathbf{t}_\nu \notin \mathcal{T}_A$ that is X -applicable on \mathcal{D}' ,
 - a) Set $\mathbf{t}'_i = \mathbf{t}_\nu, F'_i = Z'_{i-1} \cup \mathbf{op}(\mathbf{t}_\nu).$
 - b) **X-dependent step:** \cdot if $X \in \{\mathbf{bf}\text{-R}, \mathbf{bf}\text{-SO}\}$, set $Z_i = F_i.$
 \cdot if $X = \mathbf{P}$, set $Z_i = Z_{i-1}.$
 - c) Add $(\mathbf{t}'_i, F'_i, Z'_i)$ to $\mathcal{D}'.$
 - d) $i++.$
 - II) **if** \mathbf{t}_ν is the only trigger that is X -applicable on \mathcal{D}' , then
 - i) Set $\mathbf{t}'_i = \mathbf{t}_\nu, F'_i = Z'_{i-1} \cup \mathbf{op}(\mathbf{t}_\nu),$
 - ii) **if** $\mathbf{t}_\nu \in \mathcal{T}_A$, $j++.$
 - iii) **X-dependent step:** \cdot if $X \in \{\mathbf{bf}\text{-R}, \mathbf{bf}\text{-SO}\}$, set $Z_i = F_i.$
 \cdot if $X = \mathbf{P}$, set $Z_i = \widehat{F}_i.$
 - iv) Add $(\mathbf{t}'_i, F'_i, Z'_i)$ to $\mathcal{D}'.$
 - v) $i++.$
- 3) Output $\mathcal{D}'.$

The main idea behind the above algorithm is that for each rank (represented

with the k -loop of step 2) we test if we can apply the next available trigger of \mathcal{T}_A and if this test fails, we search indeterministically for other possible X-applicable triggers. In fact, we know when the test is going to fail: as established by Proposition 5.5, the rank of every trigger of \mathcal{T}_A in $\mathbb{C}(F', \mathcal{R})$ is the same as its rank in \mathcal{D} . Therefore the above algorithm will apply at each rank the corresponding triggers of \mathcal{T}_A first and then any other X-applicable triggers that are not in \mathcal{T}_A . Note that \mathcal{D}' is necessarily an X-derivation, since we only apply X-applicable triggers, and the treatment of the active factbase is in accordance with the definition of X-chase for $X \in \{\text{bf-SO}, \text{bf-R}, \text{P}\}$.

Below we prove that indeed, in every occasion where the X-applicability test of step 2.I.i fails, it is due to the breadth-first prioritization and thus there exists an X-derivation that produces A at the same rank as \mathcal{D} , starting with only the zero-rank ancestors of A in \mathcal{D} as the initial factbase. We split the proof in three cases, each with its own self-contained notation.

Case X=bf-SO: By construction, \mathcal{D}' is a bf-SO-derivation. We will verify that it does indeed include all triggers of \mathcal{T}_A . Since \mathfrak{t}_{n_1} is the first trigger in $\text{trig}(\mathcal{D})$ that produces an ancestor of A in \mathcal{D} , we know that \mathfrak{t}_{n_1} is applicable on F' , therefore it is also bf-SO-applicable on (\emptyset, F', F') and applied according to step 2.I.i of the algorithm. We will prove by contradiction, that all the triggers of \mathcal{T}_A are in $\text{trig}(\mathcal{D}')$.

Let \mathfrak{t}_{n_j} be the first trigger of \mathcal{T}_A that does not appear in \mathcal{D}' . Therefore it is SO-equivalent to a trigger \mathfrak{t}' that appears earlier in \mathcal{D}' . Suppose that $\mathfrak{t}_{n_j} = (R, \pi)$ and $\mathfrak{t}' = (R, \pi')$. The SO-equivalence guarantees that π and π' agree on the mapping of frontier variables of R . Those variables are necessarily mapped by π (so also by π') to terms of $Z^{\mathcal{D}}$. According to Algorithm 1.SO, for each rank in \mathcal{D}' , triggers of \mathcal{T}_A appear before any other trigger, so \mathfrak{t}' has to be of a strictly lower rank (in \mathcal{D}') than \mathfrak{t}_{n_j} so

$$\text{rank}_{\mathcal{D}'}(\mathfrak{t}') < \text{rank}_{\mathcal{D}'}(\mathfrak{t}_{n_j})$$

Moreover from Proposition 5.5 and Theorem 5.4

$$\text{rank}_{\mathcal{D}}(\mathfrak{t}_{n_j}) = \text{rank}_{\mathcal{D}'}(\mathfrak{t}_{n_j})$$

so we have

$$\text{rank}_{\mathcal{D}'}(\mathfrak{t}') < \text{rank}_{\mathcal{D}}(\mathfrak{t}_{n_j})$$

whereas from Theorem 5.4 and Proposition 4.11 we know that

$$\text{rank}_{(F, \mathcal{R})}(\mathfrak{t}') \leq \text{rank}_{\mathcal{D}'}(\mathfrak{t}')$$

so we can use Corollary 5.2 to conclude that there exists a retraction h from $\text{sp}(\mathfrak{t}') \cup Z^{\mathcal{D}}$ to $Z^{\mathcal{D}}$ such that $\mathfrak{t}'' = (R, h \circ \pi') \in \mathcal{R}^F$ and $\text{rank}_{(F, \mathcal{R})}(\mathfrak{t}'') \leq \text{rank}_{(F, \mathcal{R})}(\mathfrak{t}')$. Since h does not affect any variables of $Z^{\mathcal{D}}$, it does not affect the mapping of the frontier variables of R , i.e. for every $x \in \text{fr}(R)$, $h \circ \pi'(x) = \pi'(x)$. This implies that \mathfrak{t}'' is **SO**-equivalent with \mathfrak{t}' , so also with \mathfrak{t}_{n_j} . Let \mathcal{D}'' be the prefix of \mathcal{D} with all elements of rank strictly less than $\text{rank}_{\mathcal{D}}(\mathfrak{t}_{n_j})$. \mathcal{D}'' does not include \mathfrak{t}_{n_j} nor any trigger of the same **SO**-equivalence class. But $\text{sp}(\mathfrak{t}'') = h(\text{sp}(\mathfrak{t}')) \subseteq Z^{\mathcal{D}}$ and from $\text{rank}_{(F, \mathcal{R})}(\mathfrak{t}'') \leq \text{rank}_{(F, \mathcal{R})}(\mathfrak{t}')$ we know in particular that $\text{sp}(\mathfrak{t}'') \subseteq Z^{\mathcal{D}''}$. So \mathfrak{t}'' is applicable on \mathcal{D}'' . But we know that $\mathfrak{t}'' \notin \text{trig}(\mathcal{D})$ because it is **SO**-equivalent with \mathfrak{t}_{n_j} . This is a contradiction because \mathcal{D} is breadth-first, so \mathfrak{t}'' must have been applied at its respective rank.

Therefore we have shown that \mathcal{T}_A is indeed a subsequence of the sequence of triggers $\text{trig}(\mathcal{D}')$ of the **bf-SO**-derivation \mathcal{D}' from (F', \mathcal{R}) .

Case X=bf-R: By definition, \mathcal{D}' is a **bf-R**-derivation. We need to also show that \mathcal{T}_A is a subsequence of $\text{trig}(\mathcal{D}')$, i.e. that the application of new triggers inbetween does not cancel the **R**-applicability of the following the triggers of \mathcal{T}_A . We prove this by contradiction.

If there is an element of \mathcal{T}_A that does not appear in $\text{trig}(\mathcal{D}')$, there is surely the first element of \mathcal{T}_A that does not appear in $\text{trig}(\mathcal{D}')$. We assume that \mathfrak{t}_{n_j} is the first element of \mathcal{T}_A that does not appear in $\text{trig}(\mathcal{D}')$. Theorem 5.4 assures that $\text{rank}_{\mathcal{D}'}(\mathfrak{t}_{n_j}) = \text{rank}_{(F', \mathcal{R})}(\mathfrak{t}_{n_j})$ and Proposition 5.5 implies that

$rank_{(F', \mathcal{R})}(\mathfrak{t}_{n_j}) = rank_{\mathcal{D}}(\mathfrak{t}_{n_j})$ so we know that $rank_{\mathcal{D}'}(\mathfrak{t}_{n_j}) = rank_{\mathcal{D}}(\mathfrak{t}_{n_j})$. Let \mathcal{D}'' be the prefix of \mathcal{D}' with all elements of rank strictly smaller than $rank_{\mathcal{D}'}(\mathfrak{t}_{n_j})$. We denote with F'' the resulting factbase after applying all triggers of $\text{trig}(\mathcal{D}'')$ as well as any triggers that precede \mathfrak{t}_{n_j} in \mathcal{T}_A but are of the same rank as that of \mathfrak{t}_{n_j} in \mathcal{D} and in \mathcal{D}' (recall that triggers of \mathcal{T}_A have equal ranks in the two chase spaces and hence also in the two breadth-first \mathbf{R} -derivations \mathcal{D} and \mathcal{D}' as well). So it holds that

$$F'' = Z^{\mathcal{D}''} \cup \text{op}(\mathfrak{t}_{n_i}) \cup \dots \cup \text{op}(\mathfrak{t}_{n_{j-1}})$$

where the (possibly empty) set of triggers $\{\mathfrak{t}_{n_i}, \dots, \mathfrak{t}_{n_{j-1}}\}$ represents the triggers of \mathcal{T}_A which are of the same rank as \mathfrak{t}_{n_j} in \mathcal{D} .

We have assumed that \mathfrak{t} is not \mathbf{R} -applicable on F'' (since it does not appear in $\text{trig}(\mathcal{D}')$). Hence, by the condition of \mathbf{R} -applicability, there exists a homomorphism $\sigma : \text{var}(\text{op}(\mathfrak{t}_{n_j})) \rightarrow \text{term}(F'')$ with $\sigma(\text{op}(\mathfrak{t}_{n_j})) \subseteq F''$, whose domain does not include any variables from $\text{sp}(\mathfrak{t}_{n_j})$, so $\sigma(x) = x$ for every $x \in \text{var}(\text{sp}(\mathfrak{t}_{n_j}))$.

Let \mathcal{D}^j be the prefix of \mathcal{D} which includes all elements of rank strictly smaller than $rank_{\mathcal{D}}(\mathfrak{t}_{n_j})$. Notice that \mathcal{D}^j is a breadth-first \mathbf{R} -derivation and the depth of \mathcal{D}'' is equal to the depth of \mathcal{D}^j . Considering that \mathcal{D}'' corresponds to a derivation from (F, \mathcal{R}) with the same sequence of triggers and possibly lower ranks (by Proposition 4.11), we can apply the Theorem 5.3 and conclude that there exists a retraction h from $Z^{\mathcal{D}''} \cup Z^{\mathcal{D}^j}$ to $Z^{\mathcal{D}^j}$. The retraction h does not affect any variables that appear in $Z^{\mathcal{D}^j}$ therefore it does not affect any variables of $\text{sp}(\mathfrak{t}_{n_j})$ or of $\text{op}(\mathfrak{t}_{n_i}) \cup \dots \cup \text{op}(\mathfrak{t}_{n_{j-1}})$. So we have

$$h \circ \sigma(\text{op}(\mathfrak{t}_{n_j})) \subseteq h(F'')$$

which means

$$h \circ \sigma(\text{op}(\mathfrak{t}_{n_j})) \subseteq h(Z^{\mathcal{D}''} \cup \text{op}(\mathfrak{t}_{n_i}) \cup \dots \cup \text{op}(\mathfrak{t}_{n_{j-1}}))$$

and finally

$$h \circ \sigma(\mathbf{op}(\mathbf{t}_{n_j})) \subseteq Z^{\mathcal{D}^j} \cup \mathbf{op}(\mathbf{t}_{n_i}) \cup \dots \cup \mathbf{op}(\mathbf{t}_{n_{j-1}})$$

But if we denote with \mathcal{D}^* the prefix of \mathcal{D} including all triggers up to \mathbf{t}_{n_j} , we have that

$$Z^{\mathcal{D}^j} \cup \mathbf{op}(\mathbf{t}_{n_i}) \cup \dots \cup \mathbf{op}(\mathbf{t}_{n_{j-1}}) \subseteq Z^{\mathcal{D}^*}$$

therefore

$$h \circ \sigma(\mathbf{op}(\mathbf{t}_{n_j})) \subseteq Z^{\mathcal{D}^*}$$

but $\text{dom}(h) \cap Z^j = \emptyset$, thus $h \circ \sigma$ only affects new variables in $\mathbf{op}(\mathbf{t}_{n_j})$. We arrive to the conclusion that \mathbf{t}_{n_j} is not **R**-applicable on $Z^{\mathcal{D}^*}$ which is a contradiction. Hence it must be the case that \mathbf{t}_{n_j} is indeed **R**-applicable on F'' and does appear in $\text{trig}(\mathcal{D}')$. Therefore we have shown that \mathcal{T}_A is indeed a subsequence of the sequence of triggers $\text{trig}(\mathcal{D}')$ of a **bf-R**-derivation \mathcal{D}' from (F', \mathcal{R}) .

Case X=P: By definition, \mathcal{D}' is a **P**-derivation. We need to also show that \mathcal{T}_A is a subsequence of $\text{trig}(\mathcal{D}')$, i.e. that the application of new triggers inbetween does not cancel the **P**-applicability (which practically amounts to **R**-applicability on the current active factbase) of the rest of the triggers of \mathcal{T}_A . We prove this by contradiction.

If there is an element of \mathcal{T}_A that does not appear in $\text{trig}(\mathcal{D}')$, there exists surely a first element of \mathcal{T}_A that does not appear in $\text{trig}(\mathcal{D}')$. We assume that \mathbf{t}_{n_j} is the first element of \mathcal{T}_A that does not appear in $\text{trig}(\mathcal{D}')$. Let $\text{rank}_{\mathcal{D}'}(\mathbf{t}_{n_j}) = k + 1$. Theorem 5.4 assures that $\text{rank}_{\mathcal{D}'}(\mathbf{t}_{n_j}) = \text{rank}_{(F', \mathcal{R})}(\mathbf{t}_{n_j})$ and Proposition 5.5 implies that $\text{rank}_{(F', \mathcal{R})}(\mathbf{t}_{n_j}) = \text{rank}_{\mathcal{D}}(\mathbf{t}_{n_j})$, so we know that $\text{rank}_{\mathcal{D}'}(\mathbf{t}_{n_j}) = \text{rank}_{\mathcal{D}}(\mathbf{t}_{n_j})$. We denote with \mathcal{D}'^k the prefix of \mathcal{D}' with all elements of rank up to k and with \mathcal{D}^k the prefix of \mathcal{D} with all elements of rank up to k , with $Z^{\mathcal{D}'^k}$ and $Z^{\mathcal{D}^k}$ the respective final active factbases.


Let \mathcal{D}'' be the **O**-derivation from (F, \mathcal{R}) with $\text{trig}(\mathcal{D}'') = \text{trig}(\mathcal{D}'^k)$. From Proposition 4.11 and Theorem 5.4, we know that the depth of \mathcal{D}'' is smaller or equal to the depth of \mathcal{D}'^k . Note that $Z^{\mathcal{D}''} = Z^{\mathcal{D}'^k} \cup F$. Since \mathcal{D}'' and \mathcal{D}^k are both derivations in $\mathbb{C}(F, \mathcal{R})$ and \mathcal{D}^k is a breadth-first **P**-derivation, we can

apply Theorem 5.3 from which we conclude that there exists a retraction h from $Z^{\mathcal{D}'^k} \cup Z^{\mathcal{D}^k}$ to $Z^{\mathcal{D}^k}$.

We have assumed that \mathfrak{t}_{n_j} is not **R**-applicable on $Z^{\mathcal{D}'^k}$ (since it does not appear in $\text{trig}(\mathcal{D}')$). Hence, by the condition of **R**-applicability, there exists a retraction σ from $\text{op}(\mathfrak{t}_{n_j}) \cup Z^{\mathcal{D}'^k}$ to $Z^{\mathcal{D}'^k}$. The domain of σ only includes new variables of $\text{op}(\mathfrak{t}_{n_j})$. Moreover h does not affect variables of $\text{sp}(\mathfrak{t}_{n_j})$, since they appear in $Z^{\mathcal{D}^k}$. So we can compose h and σ and we have:

$$h \circ \sigma(\text{op}(\mathfrak{t}_{n_j})) \subseteq h(Z^{\mathcal{D}'^k}) \subseteq Z^{\mathcal{D}^k} \quad (5.4)$$

The substitution $h \circ \sigma$ has as domain only the set of new variables of $\text{op}(\mathfrak{t}_{n_j})$, hence it is the identity on all variables of $\text{sp}(\mathfrak{t}_{n_j})$, and from (5.4) we conclude that \mathfrak{t}_{n_j} is not **R**-applicable on $Z^{\mathcal{D}^k}$, so it cannot be in \mathcal{D} . That is a contradiction, hence it must be the case that \mathfrak{t}_{n_j} is indeed **R**-applicable on $Z^{\mathcal{D}'^k}$ and does appear in $\text{trig}(\mathcal{D}')$. Therefore we have shown that \mathcal{T}_A is indeed a subsequence of the sequence of triggers $\text{trig}(\mathcal{D}')$ of a **P**-derivation \mathcal{D}' from (F', \mathcal{R}) . \square

Corollary 5.3. **X- k -Boundedness** is decidable when $X \in \{\text{bf-O}, \text{bf-SO}, \text{bf-R}, \text{P}\}$. 

5.2.4 V-, F-, E- and C-Chase Do Not Preserve Ancestry

To conclude this section, we present examples that show that the **V**-chase, the **F**-chase, the **C**-chase and the **E**-chase do not preserve ancestry.

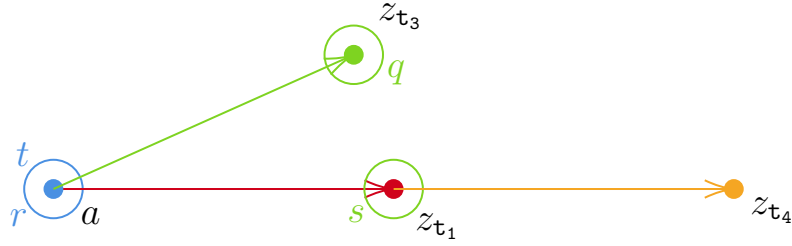
Example 36: Vacuum & Frugal Chase. Let $F = \{r(a), t(a)\}$ and \mathcal{R} the following set of rules:

$$\begin{aligned} R_1 &= r(x) \rightarrow p(x, y) \\ R_2 &= t(x) \wedge p(x, y) \rightarrow s(y) \\ R_3 &= p(x, y) \rightarrow \exists z p(x, z) \wedge q(z) \\ R_4 &= q(u) \wedge p(x, y) \rightarrow \exists z p(y, z) \end{aligned}$$

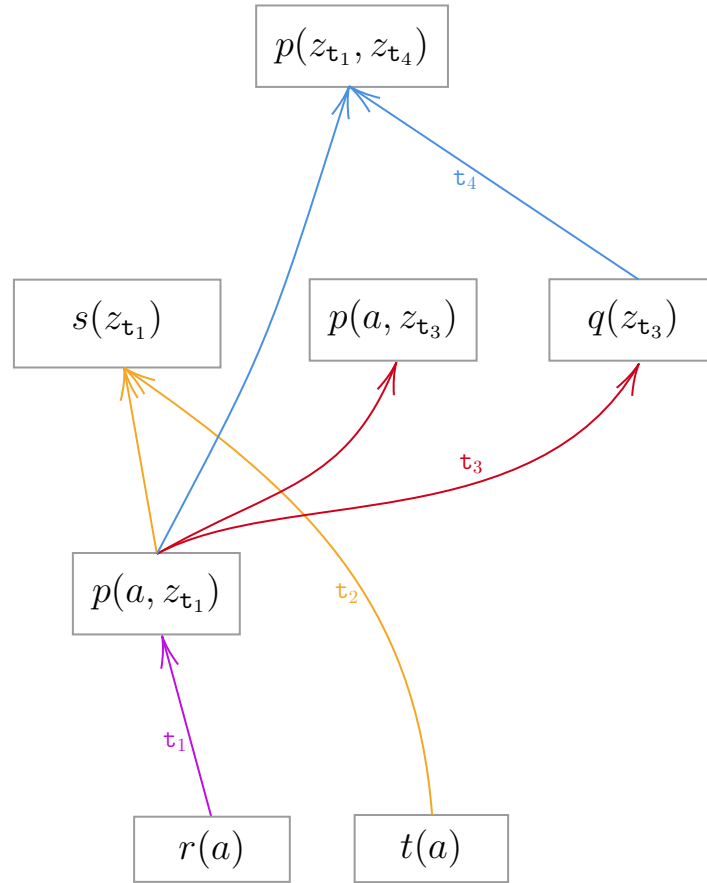
Here is a **V**- and **F**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a\})$	$F_1 = F \cup \{p(a, z_{\mathbf{t}_1})\}$	$Z_1 = F_1$	1
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_2 = F_1 \cup \{s(z_{\mathbf{t}_1})\}$	$Z_2 = F_2$	
$\mathbf{t}_3 = (R_3, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_3 = F_2 \cup \{p(a, z_{\mathbf{t}_3}), q(z_{\mathbf{t}_3})\}$	$Z_3 = F_3$	2
$\mathbf{t}_4 = (R_4, \{u \mapsto z_{\mathbf{t}_3}, x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_4 = F_3 \cup \{p(z_{\mathbf{t}_1}, z_{\mathbf{t}_4})\}$	$Z_4 = F_4$	3

We can represent $Z^{\mathcal{D}}$ as follows:



And below is the chase graph associated with \mathcal{D} :



Take the atom $A = p(z_{t_1}, z_{t_4})$. Then $\text{Anc}_{\mathcal{D}}^0(A) = \{r(a)\}$. However, starting only from $\{r(a)\}$, in a **V**- or **F**-derivation the atom $p(a, z_{t_1})$ will be removed from the active factbase with the application of t_3 , because it is isomorphically subsumed by $\{p(a, z_{t_3}), q(z_{t_3})\}$. So then t_4 will not be applicable because a part of its support, namely $p(a, z_{t_1})$, will be missing from the active factbase. Therefore the **V**-chase and the **F**-chase do not preserve ancestry. ■

Equivalent Chase: As a counterexample for the equivalent chase, we refer to the Example 41, which is used in the Section 5.5 to introduce k -minimal chase graphs.

Example 37: Core Chase. Let $F = \{r(a), t(a)\}$ and \mathcal{R} the following set of rules:

$$R_1 = r(x) \rightarrow \exists z p(x, z)$$

$$R_2 = r(x) \wedge p(x, y) \rightarrow p(x, x)$$

$$R_3 = t(x) \wedge p(x, y) \rightarrow q(y)$$

$$R_4 = p(x, y) \rightarrow \exists w p(y, w)$$

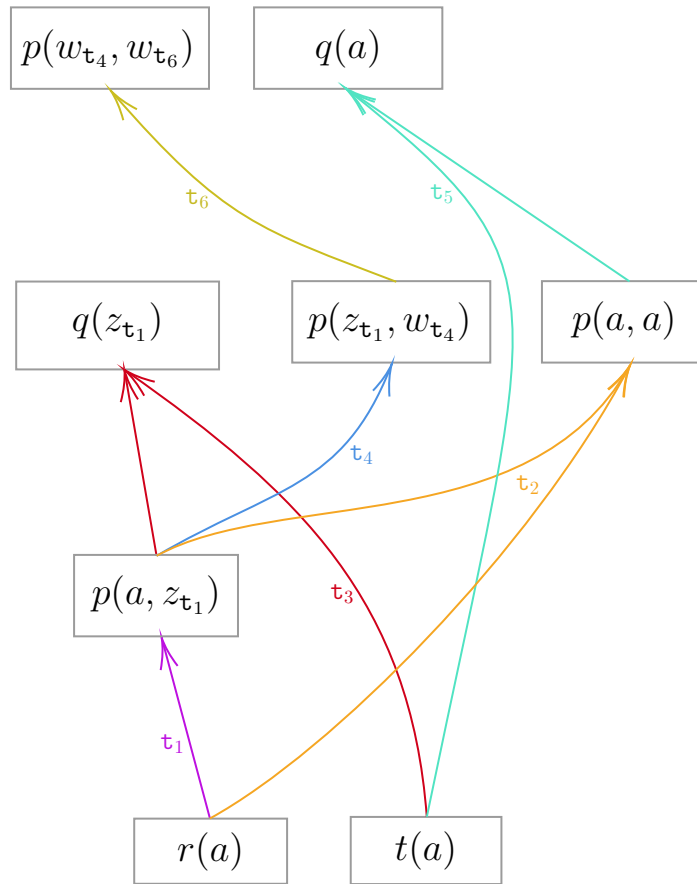
Here is a **C**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$t_1 = (R_1, \{x \mapsto a\})$	$F_1 = F \cup \{p(a, z_{t_1})\}$	$Z_1 = F_1$	1
$t_2 = (R_2, \{x \mapsto a, y \mapsto z_{t_1}\})$	$F_2 = F_1 \cup \{p(a, a)\}$	$Z_2 = Z_1$	
$t_3 = (R_3, \{x \mapsto a, y \mapsto z_{t_1}\})$	$F_3 = F_2 \cup \{q(z_{t_1})\}$	$Z_3 = Z_1$	
$t_4 = (R_4, \{x \mapsto a, y \mapsto z_{t_1}\})$	$F_4 = F_3 \cup \{p(z_{t_1}, w_{t_4})\}$	$Z_4 = Z_1 \cup F_2 \cup F_3 \cup F_4$	2
$t_5 = (R_3, \{x \mapsto a, y \mapsto a\})$	$F_5 = F_4 \cup \{q(a)\}$	$Z_5 = Z_4$	
$t_6 = (R_4, \{x \mapsto z_{t_1}, y \mapsto w_{t_4}\})$	$F_6 = F_5 \cup \{p(w_{t_4}, w_{t_6})\}$	$Z_6 = F \cup \{p(a, a), q(a)\}$	3

Below is a representation of $F^{\mathcal{D}}$:



And this is the chase graph associated with \mathcal{D} :



We take the atom $p(w_{t_4}, w_{t_6})$, and we see that its only ancestor in the initial factbase of \mathcal{D} is $r(a)$. However from the knowledge base $(\{r(a)\}, \mathcal{R})$, there does not exist a C-derivation that produces $p(w_{t_4}, w_{t_6})$. We conclude that the C-chase does not preserve ancestry. ■

5.3 Complexity Upper Bounds

The goal of this section is to investigate the complexity of the problem of deciding whether a ruleset is X - k -bounded, where X is a chase variant that preserves ancestry. Here is the precise formulation of the X - k -boundedness problem⁴:

⁴We make the usual assumption that integers are unary encoded, which implies here that the size of the encoding of k is k .

Given a ruleset \mathcal{R} and a number k in unary encoding, is it true that for every factbase F , every X-derivation from (F, \mathcal{R}) is of depth at most k ?

We will provide upper complexity bounds which follow from the implicit algorithms associated with our decidability arguments. Since our method of assuring decidability has been to bound the size of the possible factbases that we need to consider, it is implied that we do need to actually carry out the forward chaining process on these factbases. Thus to research complexity of the X- k -boundedness problem, we need to already know the complexity of constructing X-derivations of depth k .

Primarily we should discuss depth. Given a knowledge base (F, \mathcal{R}) , every exhaustive O-derivation from (F, \mathcal{R}) includes the same triggers, namely all the triggers of $\mathbb{C}(F, \mathcal{R})$. However, as shown in Example 21.0, two exhaustive O-derivations from the same knowledge base can be of different depth. That is because the order of the application of the triggers affects their ranks. Hence it is appropriate to separate the chase variants into two classes:

Definition 5.8. A chase variant X is *depth-order independent* if the existence of a k -deep X-derivation from the a knowledge base (F, \mathcal{R}) implies that every exhaustive X-derivation from (F, \mathcal{R}) is of depth at least k . A chase variant that is not depth-order independent is called *depth-order sensitive*. \dashv

A direct consequence of the above definition is that a chase variant is depth-order independent if and only if all the exhaustive X-derivations from the same knowledge base are of the same depth. When a chase variant is depth-order independent we only need to compute one X-derivation from a knowledge base to know whether there can be k -deep X-derivations from this knowledge base. Therefore the division of the chase variants into those two classes indicates that the same division takes place with respect to the complexity of the k -boundedness problem.

Proposition 5.7. The chase variants **bf-O**, **bf-SO**, **E**, **P**, **C** and **LC**⁵ are depth-order independent. The chase variants **O**, **SO**, **R**, **bf-R**, **V**, **bf-V**, **F** and **bf-F** are depth-order sensitive.

Proof: We start the proof with the depth-order sensitive chase variants. For the **O** and the **SO**-chase, the Example 21.0 serves as a counter-example to show that they are indeed depth-order sensitive. For the **R** and the **bf-R**-chase, the Example 12 serves as a counter-example. For the **V**, the **bf-V**, the **F** and the **bf-F**-chase, we modify this last example: Let \mathcal{R} be the ruleset:

$$R_1 = p(x, y) \rightarrow \exists z p(y, z)$$

$$R_2 = p(x, y) \rightarrow \exists z p(y, z) \wedge p(z, z)$$

Then with $F = \{p(a, b)\}$ we see that if we apply R_2 first, we create a terminating **V**-, **bf-V**-, **F**- and **bf-F**-derivation \mathcal{D} of depth 1. On the other hand if we apply R_1 first, then we can create a **V**-, **bf-V**-, **F**- and **bf-F**-derivation which will be of depth at least 2.

Now we know that the **bf-O**-chase is depth-order independent because in every **bf-O**-derivation \mathcal{D} of depth k , all triggers that have rank at most k in the corresponding chase space are necessarily present in $\text{trig}(\mathcal{D})$ and their ranks are equal to those of the chase space (shown in Proposition 4.5).

For the **bf-SO**-chase we refer to the Appendix (Section (B)). For the **P**-, **C**- and **LC**-chase, depth-order independence results from their definitions as synchronous derivations, which implies that the order of application of triggers does not matter. Finally to see that the **E**-chase is depth-order independent notice that for every knowledge base (F, \mathcal{R}) and every (not necessarily exhaustive) **E**-derivation \mathcal{D} from (F, \mathcal{R}) and **C**-derivation \mathcal{D}' from (F, \mathcal{R}) , it holds that if \mathcal{D} and \mathcal{D}' are of the same depth then $Z^{\mathcal{D}} \equiv Z^{\mathcal{D}'}$ (this is easy to show by induction). Hence every terminating **E**-derivation from (F, \mathcal{R}) has the same depth as any **C**-derivation from the same knowledge base. \square

Theorem 5.5. Let \mathcal{R} be a ruleset with at most b atoms in the rules' bodies and $k \in \mathbb{N}$. The number of quasi-equivalence classes of factbases of size at most b^k is in the worst case double exponential with respect to k .

⁵The **LC**-chase will be defined in the next section.

Proof: Let \mathcal{R} include p different predicates of arity at most a . As an upper bound we can assume that all the predicates have arity a . The maximum number of different variables that can be included in a factbase of size at most b^k is $a \cdot b^k$. The same holds for the maximum number of different constants. So we can say that a representative F of a quasi-equivalence class has to choose between $2a \cdot b^k$ terms for the at most $a \cdot b^k$ terms that appear in F . So if F is of size exactly b^k , we can represent it as a word of size $b^k + a \cdot b^k$, which includes the predicates' names (first) and the terms. In this case the number of possible arrangements is

$$p^{b^k} \cdot (2 \cdot a \cdot b^k)^{a \cdot b^k}$$

In our encoding we can add the possibility of the factbase having fewer atoms, by adding one more predicate (a “null” predicate), but this will only result in the replacement of p by $p + 1$, so it does not change the exponential factor. \square

Theorem 5.6. Let (F, \mathcal{R}) be a knowledge base and $k \in \mathbb{N}$. The length of a derivation from (F, \mathcal{R}) of depth k is at most double exponential with respect to k .

Proof: In the worst case scenario, we will apply all the triggers of $\mathbb{C}(F, \mathcal{R})$ so the problem is reduced to finding how many those are. Let trig^k be the set of triggers of rank at most k in $\mathbb{C}(F, \mathcal{R})$. Let b be the maximal number of atoms in the bodies of rules of \mathcal{R} and let h be the maximal number of atoms in the heads of rules of \mathcal{R} . A trigger is uniquely identified by a rule and a homomorphism. The number of potential homomorphisms from the body of a rule to the factbase is bounded by the number of permutations of the potential supports of the trigger, which include at most b atoms from F . Therefore the number of applicable triggers from \mathcal{R} on F is bounded by $|\mathcal{R}| \cdot b! \cdot |F|^b$. All these triggers will be of rank 1. Each trigger will contribute at most h atoms to the (active) factbase, so the atoms of rank at most one will be $|F| + |\mathcal{R}| \cdot b! \cdot |F|^b \cdot h$.

More generally if F^k is the output of all triggers of rank at most k , it holds that:


$$|F^k| = |F^{k-1}| + |\mathcal{R}| \cdot b! \cdot |F^{k-1}|^b \cdot h$$

which means that $|F^k| = O(|F|^{b^k})$. From this we conclude that also the number $|\mathbf{trig}^k|$ of triggers of rank at most k in $\mathbb{C}(F, \mathcal{R})$ has a similar upper bound:

$$|\mathbf{trig}^k| = \sum_{i=0}^k |\mathcal{R}| \cdot |F^i|^b = O(|F|^{b^k})$$

which shows that the length of a derivation from (F, \mathcal{R}) of depth k is double exponential with respect to k (in the worst case). \square

Now, to turn to the problem of X - k -boundedness when X preserves ancestry, suppose that \mathcal{R} is a ruleset with at most b atoms in the body of every rule. Above we showed that the number of all the possible factbases of size at most b^k (see “Ancestor Clue”-Lemma 4.3) is also double exponential with respect to k and the vocabulary of \mathcal{R} . So we can conclude that:

Corollary 5.4. Let X be a depth-order independent chase variant that preserves ancestry. The problem of determining X - k -boundedness of a ruleset \mathcal{R} is in 2-EXPTIME. 

When a chase variant is depth-order sensitive, in order to be sure whether there exists a derivation of a certain depth from a given knowledge, we will need (in the worst case) to compute all the derivations of up to this depth. Below we count how many they are:


Theorem 5.7. Let (F, \mathcal{R}) be a knowledge base and $k \in \mathbb{N}$. The number of all the derivations from (F, \mathcal{R}) of depth at most k is in the worst case triple exponential with respect to k .

Proof: An upper bound to the number of all the X -derivations from (F, \mathcal{R}) of depth at most k , is the number of all the permutations of all the triggers \mathbf{trig}^k of rank at most k in $\mathbb{C}(F, \mathcal{R})$. In Theorem 5.6 we showed that $|\mathbf{trig}^k| = O(|F|^{b^k})$. Thus

$$|\mathbf{trig}^k|! = O(|F|^{b^k})! = O\left((|F|^{b^k})^{|F|^{b^k}}\right) = O(|F|^{b^k \cdot |F|^{b^k}}) = O(2^{2^{2^k}})$$

which finally entails 3-EXPTIME as the corresponding complexity class. \square

As is the case for depth-order independent chase variants, also in depth-order sensitive chase variants, the generation of all factbases of a bounded size does not affect the upper bound of the complexity of the problem of determining k -boundedness:

Corollary 5.5. Let X be a depth-order sensitive chase variant that preserves ancestry. The problem of determining X - k -boundedness of a ruleset \mathcal{R} is in 3-EXPTIME. 

5.4 Towards The Limits of bf-R-Compliance

In this section we introduce a new chase variant that encapsulates several improvements which, following our approach up to this point, comprise in a rather evident fashion the next step in the line of research of optimizing the chase, in terms of detecting more redundancies while preserving ancestry. In the restricted chase, applicability depends on a local retraction check. This is less costly than computing a core of the whole factbase because heads of rules are usually small, but it does not trace larger scale redundancies. In the equivalent chase it is guaranteed that every rule application adds new information to the factbase, but at the (potentially great) cost of checking for logical entailment of entire factbases at every step. Lastly the core chase detects all redundancies but it does so with the computational cost of calculating the core of the factbase at every rank mark. In comparison, in our chase algorithm the trade-off between detecting redundancies and computational complexity is more balanced. We follow the paradigm of a synchronous derivation, but we calculate only a local *partial core* of the atoms at the end of a rank, by considering that the atoms of previous ranks are fixed. We will see that this is stronger than the breadth-first restricted chase in eliminating redundancy while it is also termination-order independent. The local core chase was created to push the barrier of chase variants that preserve ancestry, but as we will demonstrate, this is not the case, i.e. it does not preserve ancestry. Nonetheless, we believe that it does satisfy a very similar weaker property which guarantees the decidability of k -boundedness in

the same manner. Due to limitations of time, we chose not to undertake the research for a proof of this conjecture.

5.4.1 Local Core Chase

We want to specify minimal retracts of an atomset F that do not affect variables from a predefined variable set W . To this end we define what we call a *partial core*.

Definition 5.9 (Partial Core). Let F be an atomset and $W \subseteq \mathbf{var}(F)$. A set $F' \subseteq F$ is a *partial core* of F *preserving* W , which we denote by $pcore(F, W)$, if and only if the following two hold:

- i) F' is a retract of F resulting from a retraction σ that does not affect any of the variables in W , i.e. $F' = \sigma(F)$ with $dom(\sigma) \cap W = \emptyset$.
- ii) σ is minimal for the above property, i.e. every non-trivial retraction from F' affects at least one variable from W . ⊥

To further establish the notion of partial core, we devote a part of the Appendix (Section (C)) to discuss the *freezing* and *unfreezing* operations on variables of an atomset. Based on the notion of partial core, we are now ready to define the new chase variant. We remind that in a synchronous derivation \mathcal{D} , if D_i is a rank mark, we denote with \widehat{F}_i the union of the transitory factbases of the current (ending) rank.

Definition 5.10 (Local Core Chase). A **local core derivation** is any synchronous derivation $\mathcal{D} = (\mathbf{t}_*, F_*, Z_*)$ from (F, \mathcal{R}) where for every rank mark D_i with $i > 0$, the active factbase is $Z_i = pcore(\widehat{F}_i, W)$, where $W = \{x \in \mathbf{var}(F^{\mathcal{D}}) \mid rank(x) < rank(D_i)\}$. ⊥

We will abbreviate local core chase with **LC-chase**. By the definition, we know that the **LC-chase** is termination-order independent because it comprises synchronous derivations with a retraction on every rank mark which is indifferent to the ordering of the triggers that preceded.

Example 38: Let $F = \{r(a), t(a)\}$ and \mathcal{R} the following set of rules:

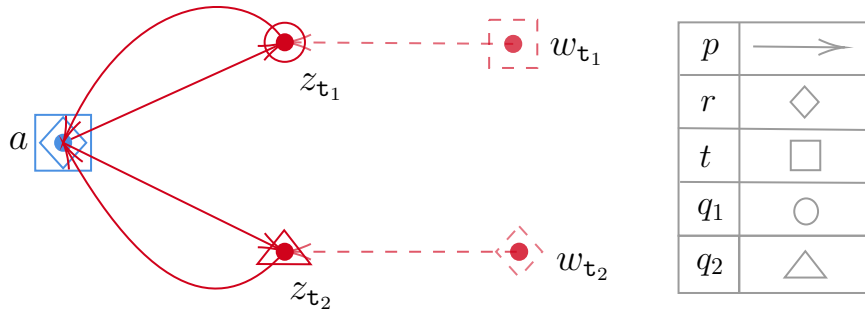
$$R_1 = r(x) \rightarrow \exists z \exists w p(x, z) \wedge p(z, x) \wedge q_1(z) \wedge p(w, z) \wedge t(w)$$

$$R_2 = t(x) \rightarrow \exists z \exists w p(x, z) \wedge p(z, x) \wedge q_2(z) \wedge p(w, z) \wedge r(w)$$

Here is a **LC**-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a\})$	$F_1 = F \cup \{p(a, z_{\mathbf{t}_1}), p(z_{\mathbf{t}_1}, a), q_1(z_{\mathbf{t}_1}), p(w_{\mathbf{t}_1}, z_{\mathbf{t}_1}), t(w_{\mathbf{t}_1})\}$	$Z_1 = F$	
$\mathbf{t}_2 = (R_2, \{x \mapsto a\})$	$F_2 = F \cup \{p(a, z_{\mathbf{t}_2}), p(z_{\mathbf{t}_2}, a), q_2(z_{\mathbf{t}_2}), p(w_{\mathbf{t}_2}, z_{\mathbf{t}_2}), r(w_{\mathbf{t}_2})\}$	$Z_2 = F \cup \{p(a, z_{\mathbf{t}_1}), p(a, z_{\mathbf{t}_2}), p(z_{\mathbf{t}_1}, a), p(z_{\mathbf{t}_2}, a), q_1(z_{\mathbf{t}_1}), q_2(z_{\mathbf{t}_2})\}$	1

Here is a representation of $F^{\mathcal{D}}$, where the dashed elements do not appear in $Z^{\mathcal{D}}$:



We see that there is no **R**-applicable trigger on $Z^{\mathcal{D}}$, so \mathcal{D} is terminating. On the other hand, we can verify that when $X \in \{\mathbf{O}, \mathbf{SO}, \mathbf{R}, \mathbf{P}, \mathbf{V}, \mathbf{F}\}$, every exhaustive **X**-derivation is infinite. ■

The local core chase is by definition stronger than the parallel chase in detecting redundancies. But as it is a breadth-first algorithm, it fails to detect redundancies of the type that we saw at Example 22, where we showed that the breadth-first strategy is not always optimal for the restricted chase. However, it is easy to see that the local core chase reflects the termination of the breadth-first restricted chase. And even more, if there exists a terminating **bf-R**-derivation from (F, \mathcal{R}) , then every exhaustive **LC**-derivation from (F, \mathcal{R}) is terminating.

We proceed now to show that the **LC**-chase does not preserve ancestry.

Example “LCANCESTRY” 39: Let $F = \{q_1(b, a), q_2(c, a), s(a)\}$ and \mathcal{R} is:

$$R_1 = q_1(x, y) \rightarrow \exists z \exists w \exists v p(y, z) \wedge p(w, z) \wedge s(w) \wedge q_1(v, w)$$

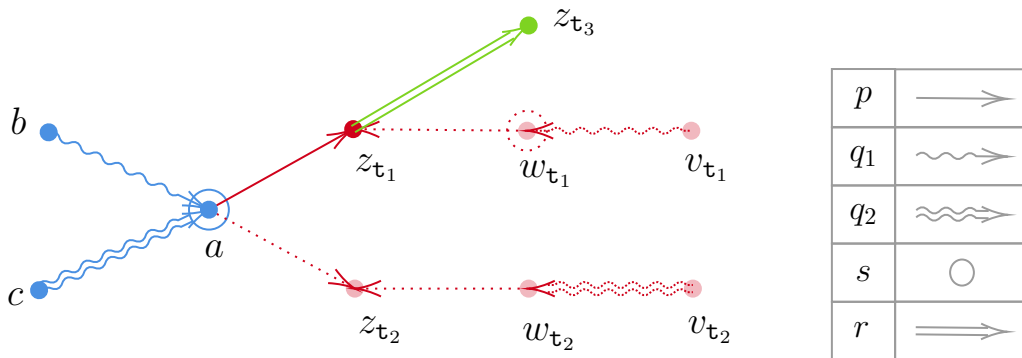
$$R_2 = s(x) \rightarrow \exists z \exists w \exists v p(x, z) \wedge p(w, z) \wedge q_2(v, w)$$

$$R_3 = s(x) \wedge p(x, y) \rightarrow \exists z r(y, z)$$

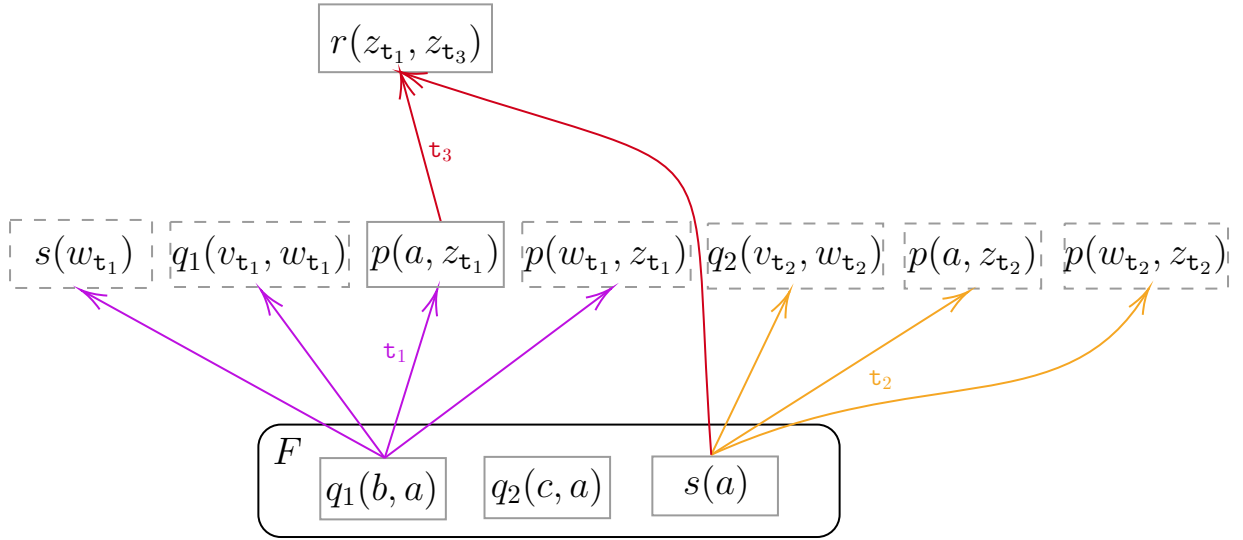
Below is the derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto b, y \mapsto a\})$	$F_1 = F \cup \{p(a, z_{\mathbf{t}_1}), p(w_{\mathbf{t}_1}, z_{\mathbf{t}_1}), s(w_{\mathbf{t}_1}), q_1(v_{\mathbf{t}_1}, w_{\mathbf{t}_1})\}$	$Z_1 = F$	
$\mathbf{t}_2 = (R_2, \{x \mapsto a\})$	$F_2 = F \cup \{p(a, z_{\mathbf{t}_2}), p(w_{\mathbf{t}_2}, z_{\mathbf{t}_2}), q_2(v_{\mathbf{t}_2}, w_{\mathbf{t}_2})\}$	$Z_2 = F \cup \{p(a, z_{\mathbf{t}_1})\}$	1
$\mathbf{t}_3 = (R_3, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_3 = Z_2 \cup \{r(z_{\mathbf{t}_1}, z_{\mathbf{t}_3})\}$	$Z_3 = F \cup \{r(z_{\mathbf{t}_1}, z_{\mathbf{t}_3})\}$	2

\mathcal{D} is a terminating LC-chase derivation. Notice that when selecting the partial core in Z_2 , we chose $p(a, z_{\mathbf{t}_1})$ over $p(a, z_{\mathbf{t}_2})$. This choice has no semantic impact on the derivation. However it affects its chase graph, so also the ancestors/descendants of the certain atoms. Here is $F^{\mathcal{D}}$ where the atoms that do not appear in $Z^{\mathcal{D}}$ are dotted:



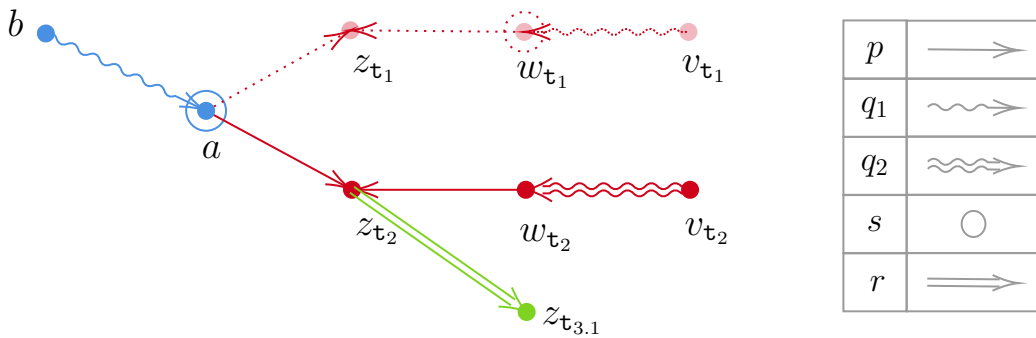
And here is the chase graph associated with \mathcal{D} :



We have that $\text{Anc}_{\mathcal{D}}^0(r(z_{t_1}, z_{t_3})) = \{q_1(b, a), s(a)\}$. Below is the only terminating LC-derivation from (F', \mathcal{R}) , where $F' = \{q_1(b, a), s(a)\}$, which we call \mathcal{D}' :

\emptyset	$F'_0 = \{q_1(b, a), s(a)\}$	$Z'_0 = F'_0$	0
$t_1 = (R_1, \{x \mapsto b, y \mapsto a\})$	$F'_1 = F'_0 \cup \{p(a, z_{t_1}), p(w_{t_1}, z_{t_1}), s(w_{t_1}), q_1(v_{t_1}, w_{t_1})\}$	$Z'_1 = Z'_0$	
$t_2 = (R_2, \{x \mapsto a\})$	$F'_2 = F'_0 \cup \{p(a, z_{t_2}), p(w_{t_2}, z_{t_2}), q_2(v_{t_2}, w_{t_2})\}$	$Z'_2 = Z'_0 \cup \{p(a, z_{t_2}), p(w_{t_2}, z_{t_2}), q_2(v_{t_2}, w_{t_2})\}$	1
$t_{3.1} = (R_3, \{x \mapsto a, y \mapsto z_{t_2}\})$	$F'_3 = Z'_2 \cup \{r(z_{t_2}, z_{t_{3.1}})\}$	$Z'_3 = Z'_2 \cup \{r(z_{t_2}, z_{t_{3.1}})\}$	2

Below is $F'^{\mathcal{D}'}$ where the atoms that do not appear in $Z'^{\mathcal{D}'}$ are dotted:



We see that the atom $r(z_{t_1}, z_{t_3})$ is not produced in \mathcal{D}' . Therefore the LC-chase does not preserve ancestry. ■

5.4.2 The Conjecture

As mentioned in the beginning of this section, our aim when defining the **LC**-chase was to obtain a chase variant that is stronger in eliminating redundancy than the **bf-R**-chase while retaining the property of preservation of ancestry. The above counter-example did show that this is indeed not the case, but there is a reason to suspect that this finding does not kill the project. As we pointed out in the above example, the non-deterministic choice of (partial) core influences the ancestor/descendant relations in the resulting **LC**-derivation. In other words, although all the **LC**-derivations from a given knowledge base produce isomorphic results for every rank, the corresponding **LC**-chase graphs are not isomorphic. Our presumption is that for each atom in a given active fact-base, there exists at least one of those **LC**-chase graphs that has a corresponding atom that preserves ancestry, meaning that we can indeed reproduce the same atom from the same rank starting from only its ancestors. If this is the case, then a weaker form of preservation of ancestry holds for the **LC**-chase, leading to the decidability of k -boundedness in exactly the same way as preservation of ancestry does.

Our work on the subject indicates that a confirmation of this hypothesis is feasible but outside of the time limits for this thesis. Therefore we will only formulate the weaker property that we suggest that the **LC**-chase satisfies, leaving the potential proof for future work. We begin by specifying what are *isomorphic triggers*:

Definition 5.11. Let $\tau = (R, \pi)$ and $\tau' = (R, \pi')$ be two triggers. We say that τ and τ' are *isomorphic* if there is a variable renaming σ with $\text{dom}(\sigma) \subseteq \text{codom}(\pi)$ and $\text{codom}(\sigma) \cap \text{codom}(\pi) = \emptyset$, such that $\pi' = \sigma \circ \pi$. Let $\dot{\sigma}$ be the extension of σ created by adding the corresponding existential variable mappings ($x_\tau \mapsto x_{\tau'}$). We say that τ is isomorphic with τ' by $\dot{\sigma}$. \dashv

As a result of the above definition, if τ is isomorphic with τ' by $\dot{\sigma}$, then it holds that $\dot{\sigma}(\text{op}(\tau)) = \text{op}(\tau')$ and $\dot{\sigma}(\text{sp}(\tau)) = \text{sp}(\tau')$.

Now we need to introduce a notion of isomorphic derivations. In this case, the “isomorphism” is not going to be one mapping, but a sequence of mappings,

between the elements of the derivations (focusing on triggers and active fact-bases), which will ensure that the two derivations do in fact evolve in exactly the same manner.

Definition 5.12. Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ and $\mathcal{D}' = (\mathfrak{t}'_*, F'_*, Z'_*)$ be two derivations from (F, \mathcal{R}) . We say that \mathcal{D} and \mathcal{D}' are *isomorphic* if they are of the same length n (possibly infinite) and there exists a sequence of pairs of variable renamings $(\sigma_0, \tau_0), (\sigma_1, \tau_1), \dots, (\sigma_n, \tau_n)$ such that

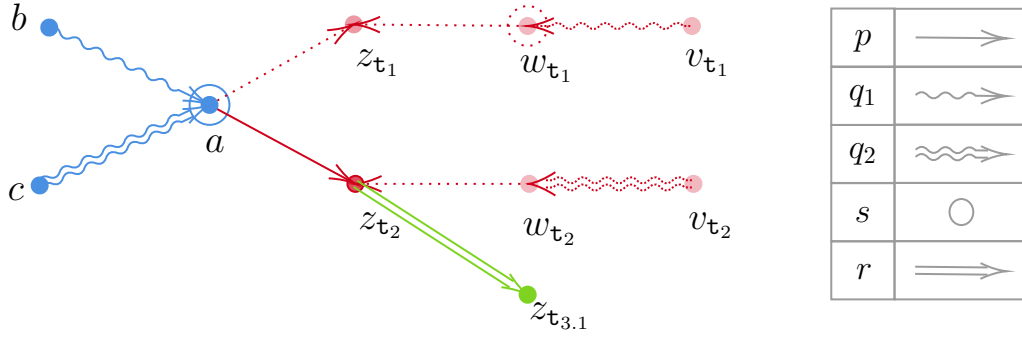
- for every $i \leq n$ it holds that \mathfrak{t}_i is isomorphic with \mathfrak{t}'_i by σ_i .
- $\tau_0 \subseteq \tau_1 \subseteq \tau_2 \subseteq \dots \subseteq \tau_n$, where the substitutions are seen as sets of mappings.
- for every $i \leq n$ it holds that $Z'_i = \tau_i(Z_i)$.

When \mathcal{D} and \mathcal{D}' are isomorphic we say that an atom A produced by \mathfrak{t}_i in \mathcal{D} *corresponds* to the atom $\sigma_i(A)$ produced by \mathfrak{t}'_i in \mathcal{D}' (and vice versa). \dashv

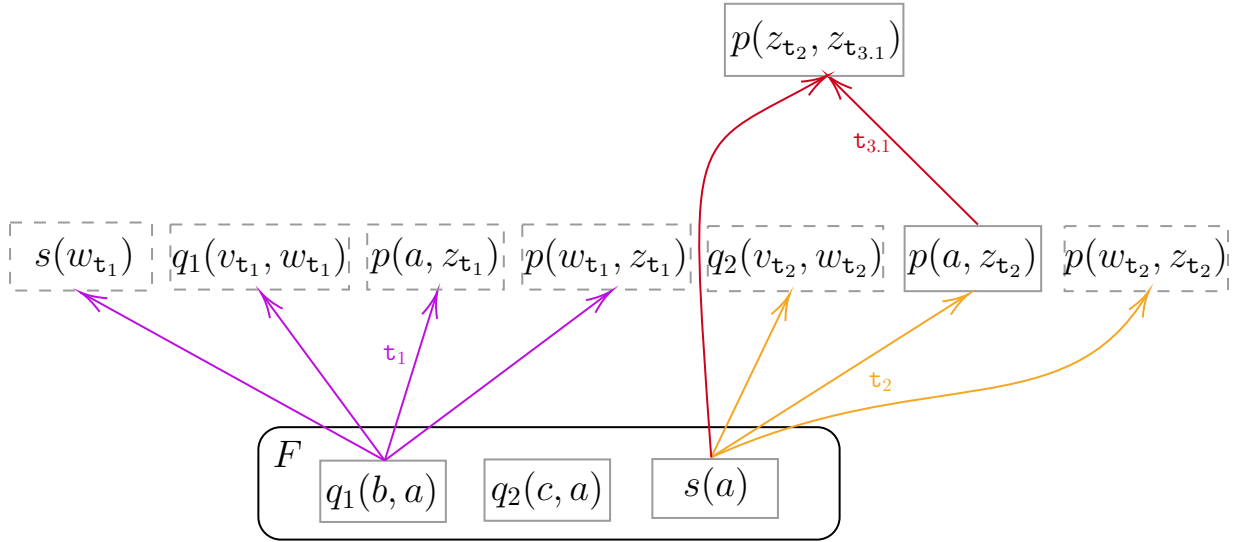
Example “LCANCESTRY” 40 (continued from Example 39): In the previous subsection we specified the knowledge base (F, \mathcal{R}) and a LC-derivation \mathcal{D} from (F, \mathcal{R}) . We now specify the derivation \mathcal{D}'' from (F, \mathcal{R}) which is isomorphic with \mathcal{D} :

\emptyset	$F_0 = F$	$Z_0 = F_0$	0
$\mathfrak{t}_1 = (R_1, \{x \mapsto b, y \mapsto a\})$	$F_1 = F \cup \{p(a, z_{\mathfrak{t}_1}), p(w_{\mathfrak{t}_1}, z_{\mathfrak{t}_1}), s(w_{\mathfrak{t}_1}), q_1(v_{\mathfrak{t}_1}, w_{\mathfrak{t}_1})\}$	$Z_1 = F$	
$\mathfrak{t}_2 = (R_2, \{x \mapsto a\})$	$F_2 = F \cup \{p(a, z_{\mathfrak{t}_2}), p(w_{\mathfrak{t}_2}, z_{\mathfrak{t}_2}), q_2(v_{\mathfrak{t}_2}, w_{\mathfrak{t}_2})\}$	$Z_2'' = F \cup \{p(a, z_{\mathfrak{t}_2})\}$	1
$\mathfrak{t}_{3.1} = (R_3, \{x \mapsto a, y \mapsto z_{\mathfrak{t}_2}\})$	$F_3'' = Z_2'' \cup \{r(z_{\mathfrak{t}_2}, z_{\mathfrak{t}_{3.1}})\}$	$Z_3'' = F \cup \{r(z_{\mathfrak{t}_2}, z_{\mathfrak{t}_{3.1}})\}$	2

Next, we provide an illustration of $F^{\mathcal{D}''}$, where the atoms that do not appear in $Z^{\mathcal{D}''}$ are dotted:



As it is expected it holds that $F^{\mathcal{D}}$ is isomorphic with $F^{\mathcal{D}''}$ and $Z^{\mathcal{D}}$ is isomorphic with $Z^{\mathcal{D}''}$. However the associated chase graphs are not isomorphic. Indeed, here is the LC-chase graph associated with \mathcal{D}'' :



Notice that the atom $p(z_{t_1}, z_{t_3})$ in \mathcal{D} corresponds to the atom $p(z_{t_2}, z_{t_{3.1}})$ in \mathcal{D}'' . We have that $\text{Anc}_{\mathcal{D}''}^0(r(z_{t_2}, z_{t_{3.1}})) = \{s(a)\}$. And in this case we can see that every exhaustive LC-derivation from $(\{s(a)\}, \mathcal{R})$ produces $p(z_{t_2}, z_{t_{3.1}})$ in the same rank as \mathcal{D}'' . ■

As we saw in the above example, the choice of partial core during the local core chase affects whether our derivation will comply with the preservation of ancestry. All the LC-derivations from the same knowledge base are isomorphic. Our conjecture is, that there should always be one derivation, where the choice of partial cores is such that ancestry is preserved. This assertion motivates the definition of another property that will account for those choices.

Definition 5.13 (Loose Preservation of Ancestry). The X-chase is said to *loosely preserve ancestry* if, for every X-derivation \mathcal{D}_1 from (F, \mathcal{R}) , for every atom A in $F^{\mathcal{D}_1}$, there exists an isomorphic X-derivation \mathcal{D}_2 , an atom A' in $F^{\mathcal{D}_2}$ corresponding to A and an X-derivation \mathcal{D}'_2 from $(\text{Anc}_{\mathcal{D}_2}^0(A'), \mathcal{R})$ such that A' is produced in \mathcal{D}'_2 and $\text{rank}_{\mathcal{D}_2}(A') = \text{rank}_{\mathcal{D}'_2}(A')$. \dashv

In Theorem 5.1 we showed that preservation of ancestry implies the decidability of k -boundedness. The proof of this theorem can be modified (in a rather trivial fashion) so as to show that loose preservation of ancestry also implies decidability of k -boundedness. We believe that the LC-chase loosely preserves ancestry. If our conjecture is true, then k -LC-boundedness is decidable.

5.5 k -Minimal Chase Graphs

In this section we link boundedness and k -boundedness with a certain type of chase graphs, called *k -minimal chase graphs*. In this way we arrive at an abstract characterization of k -boundedness, which turns to an algorithmic characterization for chase variants that preserve ancestry since in this case we can compute the k -minimal chase graphs. For the rest of the chase variants it remains an open problem whether we can compute k -minimal chase graphs. A positive solution to this problem assures decidability of k -boundedness.

5.5.1 A Characterization of k -Boundedness

Here we examine the question, given a ruleset \mathcal{R} and an atomset S , is there a factbase such that there is an X-derivation on (F, \mathcal{R}) that produces an isomorphic atomset \bar{S} at rank $k + 1$? This is important because it can be used to show that X-derivations with \mathcal{R} can achieve depth equal to $k + 1$, therefore \mathcal{R} is not X- k -bounded and if it is X-bounded, the bound will have to be more than k .

If there is such a factbase, then surely there is a minimal one. Based on this observation, and using the chase graph framework to facilitate the view on rank/depth of derivations, we introduce the following concept:

Definition 5.14. Let \mathcal{R} be an ontology and S an atomset. An X-chase graph $G = (V, E)$ on (F, \mathcal{R}) is called *k-minimal* for S with \mathcal{R} if:

- i. there is an atomset \bar{S} isomorphic with S such that $\bar{S} \in V^k$ and
- ii. for all $F' \subset F$, for every X-chase graph $G' = (V', E')$ on (F', \mathcal{R}) , holds that $\bar{S} \not\subseteq V'^k$.
- iii. there does not exist an X-chase graph G'' on (F, \mathcal{R}) that satisfies the conditions i and ii and is a strict subgraph of G .

The set of all X-chase graphs (equivalence classes up to quasi-isomorphism) that are *k-minimal* for S with \mathcal{R} is denoted with $\mathbf{B}_X^k(S, \mathcal{R})$. \dashv

The “B” notation stems from the remark that we are actually traversing towards the concept of backward chaining here. Indeed, if we can find a way to compute *k-minimal* X-chase graphs for an atomset, then we will have reverse-engineered the forward chaining process.

Example 41: 2-minimal E-chase graph. Let \mathcal{R} be the following set of rules:

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

$$R_2 = s(x, y) \rightarrow p(x, y)$$

$$R_3 = t(x) \wedge p(x, y) \rightarrow q(y)$$

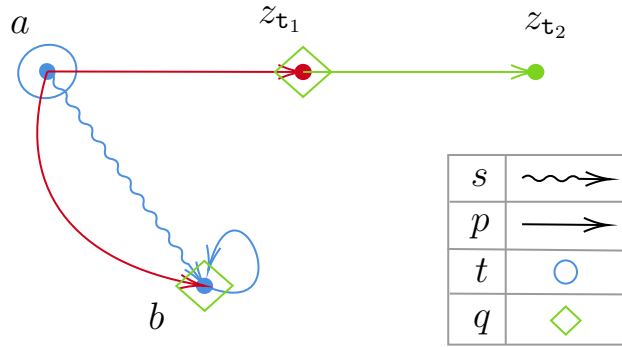
$$R_4 = p(x, y) \rightarrow \exists w p(y, w)$$

Suppose that we are looking for a 2-minimal E-chase graph with \mathcal{R} , for the atomset $\{p(x, y)\}$. In other words, we are looking for an E-derivation with \mathcal{R} which produces an atom A isomorphic to $p(x, y)$ at rank 2, with the extra property that if we remove any atom from the initial factbase, we loose A from the atoms produced at rank 2.

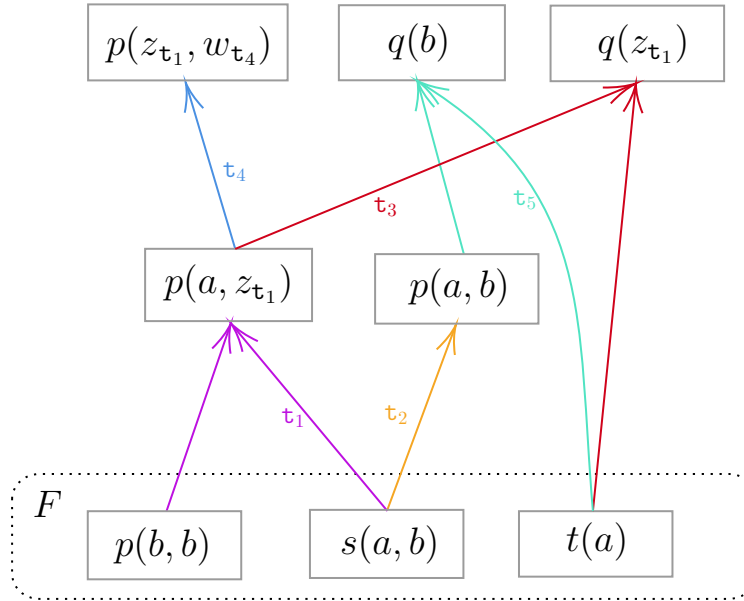
Let $F = \{s(a, b), p(b, b), t(a)\}$. Here is an E-derivation \mathcal{D} from (F, \mathcal{R}) :

\emptyset	F	$Z_0 = F$	0
$\mathbf{t}_1 = (R_1, \{x \mapsto a, y \mapsto b\})$	$F_1 = F \cup \{p(a, z_{\mathbf{t}_1})\}$	$Z_1 = F_1$	
$\mathbf{t}_2 = (R_2, \{x \mapsto a, y \mapsto b\})$	$F_2 = F_1 \cup \{p(a, b)\}$	$Z_2 = F_2$	1
$\mathbf{t}_3 = (R_3, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_3 = F_2 \cup \{q(z_{\mathbf{t}_1})\}$	$Z_3 = F_3$	
$\mathbf{t}_4 = (R_4, \{x \mapsto a, y \mapsto z_{\mathbf{t}_1}\})$	$F_4 = F_3 \cup \{p(z_{\mathbf{t}_1}, w_{\mathbf{t}_4})\}$	$Z_4 = F_4$	
$\mathbf{t}_5 = (R_3, \{x \mapsto a, y \mapsto b\})$	$F_5 = F_4 \cup \{q(b)\}$	$Z_5 = F_5$	2

Below we find the graphical representation of Z_5 :



And here is the **E**-chase graph G associated with \mathcal{D} :



We can verify that G is 2-minimal for $p(x, y)$ with \mathcal{R} , since $p(z_{\mathbf{t}_1}, w_{\mathbf{t}_4})$ is isomorphic with $p(x, y)$ and if we reduce F by any way, the (**E**-)application of trigger \mathbf{t}_4 at rank 2 will be impossible. Of course G is not the only 2-minimal

E-chase graph for $p(x, y)$ with \mathcal{R} , but it is the one which is interesting to investigate. This derivation also serves as a counterexample to show that the **E**-chase does not preserve ancestry: note that $\text{Anc}_{\mathcal{D}}^0(p(z_{t_1}, w_{t_4})) = \{s(a, b), p(b, b)\}$. But starting only from $\text{Anc}_{\mathcal{D}}^0(p(z_{t_1}, w_{t_4}))$, there is no **E**-derivation that produces $p(z_{t_1}, w_{t_4})$, because t_4 is not **E**-applicable. Hence we can say that the **E**-chase does not preserve ancestry. ■

Next, we employ this new notion to characterize **X**-boundedness. Notice that up to quasi-isomorphism there are a finite number of triggers associated with every rule of a ruleset. So if a ruleset can produce derivations of depth more than k , at least one of those triggers appears at rank k . In particular, a subset S of its output will comprise an atomset which is entirely of rank k . So the atomset S will have a k -minimal **X**-chase graph.

We remind that given an atomset S , a *specialization* of S is any atomset $\pi(S)$, which is the image of a substitution π on S .

Theorem 5.8. Let \mathcal{R} be a ruleset. \mathcal{R} is **X**-bounded if and only if for every $R = (B, H) \in \mathcal{R}$ and for every subset S of a specialization $\pi(H)$ of the head of R , there is $k \in \mathbb{N}$ such that $\mathbf{B}_X^k(S, \mathcal{R}) = \emptyset$.

Proof: If \mathcal{R} is **X**-bounded then there exists a $k \in \mathbb{N}$ such that for every F , every **X**-chase graph on (F, \mathcal{R}) is of depth strictly less than k , hence $\mathbf{B}_X^k(S, \mathcal{R}) = \emptyset$ for any atomset S . So also for $S \subseteq \pi(H)$, where π is any substitution and H is the head of any rule from \mathcal{R} .

Suppose that for every set S that is a subset of a specialization $\pi(H)$ of the head of a rule $R \in \mathcal{R}$, there is $k \in \mathbb{N}$ such that $\mathbf{B}_X^k(S, \mathcal{R}) = \emptyset$. Let k' be the maximum bound for all such sets (they are finite up to quasi-isomorphism). So it has to be that for any rule R and any subset S of a specialization of the head of R holds that $\mathbf{B}_X^{k'}(S, \mathcal{R}) = \emptyset$.

We assume that \mathcal{R} is not **X**-bounded. Then there is a factbase F and an **X**-chase graph $G = (V, E)$ on (F, \mathcal{R}) , such that the depth of G is k' . Then there is some rule application made at rank k' in G . So there is a substitution π , a rule $R = (B, H) \in \mathcal{R}$ and an atom $A \in \pi^s(H)$ with $A \in V^{k'}$. If G is not minimal wrt $\{A\}$, then there exists a minimal k' -deep **X**-chase graph $G' = (U, E')$ on

(F', \mathcal{R}) such that $A \in U^{k'}$. Then $G' \in \mathbf{B}_X^{k'}(\{A\}, \mathcal{R})$ which is a contradiction, hence there is no k' -deep X-chase graph on (F, \mathcal{R}) , for any F . So \mathcal{R} is bounded by k' . \square

From Theorem 5.8 we obtain a characterization of k -boundedness:

Corollary 5.6. A ruleset \mathcal{R} is X- k -bounded if and only if for every $R = (B, H) \in \mathcal{R}$ and for every subset S of a specialization $\pi(H)$ we have that $\mathbf{B}_X^k(S, \mathcal{R}) = \emptyset$. \clubsuit

We deduce that the research on $(k\text{-})$ X-boundedness would benefit if there was an algorithm to generate k -minimal X-chase graphs. We have constructed such an algorithm for the chase variants that preserve ancestry (see next subsection), while for other chase variants it is an open question whether there exists such an algorithm. Even more, it is not clear if the set $\mathbf{B}_X^k(S, \mathcal{R})$ is always finite, i.e. it is possible that for some chase variants there exists a ruleset \mathcal{R} , an atomset S and a number k such that the set of all (S, k) -minimal X-chase graphs with \mathcal{R} is infinite (even considering equivalence classes up to quasi-isomorphism).

5.5.2 Computing k -Minimal X-Chase Graphs

The goal of this section is to examine whether we can compute all possible k -minimal X-chase graphs for some given atomset S with a ruleset \mathcal{R} . We will show that this is achievable for any chase which preserves ancestry. When a chase variant does not preserve ancestry, namely for $X \in \{\mathbf{E}, \mathbf{C}, \mathbf{F}\}$, we can trace the possible ancestors of a given atomset in some derivation, but it remains to be seen if this can be used to compute k -minimal X-chase graphs.

We define the notion of *generator* of an atomset S with a rule R . Our perspective is an abstraction of the concept of piece-based query rewriting [41, 44] (details in Appendix Section (A)).

Definition 5.15. Let \mathcal{R} be a ruleset and S be an atomset. Let $\{S_1, S_2, \dots, S_n\}$ be a partition of S such that there exist triggers $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n$ from \mathcal{R} and a (bijective) renaming

$$\sigma : \mathbf{nul}(\mathbf{op}(\mathbf{t}_1) \cup \dots \cup \mathbf{op}(\mathbf{t}_n)) \rightarrow \mathbf{var}(S)$$

where it holds that

$$\sigma(\mathbf{op}(\mathbf{t}_1) \cup \dots \cup \mathbf{op}(\mathbf{t}_n)) = S$$

Then the atomset $\mathbf{sp}(\mathbf{t}_1) \cup \mathbf{sp}(\mathbf{t}_2) \cup \dots \cup \mathbf{sp}(\mathbf{t}_n)$ is a *generator* of S with \mathcal{R} . \dashv

The intuition behind the above definition is simple: a generator of an atomset S with a ruleset \mathcal{R} is any factbase from which we can produce S with the parallel application of some triggers from \mathcal{R} , i.e. in one breadth-first step.

Lemma 5.1. The set of all generators of an atomset S is finite (up to quasi-isomorphism).

Proof: The number of partitions of S is finite. Furthermore for each given atomset $S' \subseteq S$, the number of mappings $\sigma \circ \pi$ from the head of a rule $R = (B, H)$ in \mathcal{R} such that $\sigma \circ \pi(H) = S'$ where σ is a renaming with $\mathbf{dom}(\sigma) = \mathbf{exv}(R)$ and $\mathbf{dom}(\pi) = \mathbf{fr}(R)$ are also finite. But then we can remove the existential variable mappings from π and extend it so that it also maps the disappearing⁶ variables of R to any terms. There are infinite such extensions of π but only finite up to quasi-isomorphism of $\pi(B)$. So there finitely many possible triggers $\mathbf{t} = (R, \pi)$ with the desired properties. \square

In Subsection 4.5.3 we defined the notion of generator of an atom in a chase space, as any maximal set of direct ancestors of this atom with edges of the same label (trigger). This corresponds to the more general notion that we define here. In a chase graph (which is a subgraph of the respective chase space) the set of direct ancestors of an atom is necessarily connected with edges of the same label, which is the trigger that produced this atom. The following proposition supports this connection and generalizes for atomsets instead of just one atom:

⁶i.e. the variables that appear in the body but not in the head of R .

Proposition 5.8. Let $G = (V, E)$ be a k -deep X-chase graph on (F, \mathcal{R}) and $S \subseteq V$. The set of direct ancestors of S in G is a generator of S .

Proof: This results from Definition 5.15, since a generator of S is the union of the supports of the triggers that produce S . \square

By definition there is a connection between generator and trigger. Proposition 5.8 goes a step further, by linking the notion of generator with that of direct ancestors in a possible derivation/chase graph. Therefore we can argue that we have obtained an inverse view of the rule application process. A consequent research goal is therefore to expand this inverse view to the whole forward chaining process. Utilizing the notion of k -minimal chase graphs, we will show that this is certainly achievable in chase variants that preserve ancestry. It remains to be seen whether the rest of the chase variants can be similarly reversely engineered.

Definition 5.16 (Ancestor Trees). Let \mathcal{R} be a ruleset, S an atomset, θ a variable renaming and $G = (V, E)$ a chase graph of depth k on (V^0, \mathcal{R}) such that $\theta(S) \subseteq V^k$. The subgraph of G induced by $\theta(S) \cup \text{Anc}_G(\theta(S))$, is an *ancestor tree* of S with \mathcal{R} . \dashv

Although we name it *tree*, an ancestor tree is usually not a tree graph. Rather it is comprised of multiple trees in superposition, with the atoms of $\theta(S)$ as roots. Moreover the direction of the edges is from the leaves to the root(s). Two ancestor trees are *equivalent* if they are quasi-isomorphic. We will use the symbol \mathbf{T} for an equivalence class of ancestor trees and we will usually identify an ancestor tree with its equivalence class. The *depth* of an ancestor tree \mathbf{T} is the maximum length of a path in \mathbf{T} . We will denote the set of all (equivalence classes of) ancestor trees of an atomset S with a ruleset \mathcal{R} of depth k with $\mathbf{T}(S, \mathcal{R}, k)$. So we can naturally also denote with $\mathbf{T}(S, \mathcal{R}) = \bigcup_{k \in \mathbb{N}} \mathbf{T}(S, \mathcal{R}, k)$ the set of all ancestor trees of S with \mathcal{R} .

Remark 5.9. For every atomset S , ruleset \mathcal{R} and number k , the set $\mathbf{T}(S, \mathcal{R}, k)$ is finite. \clubsuit

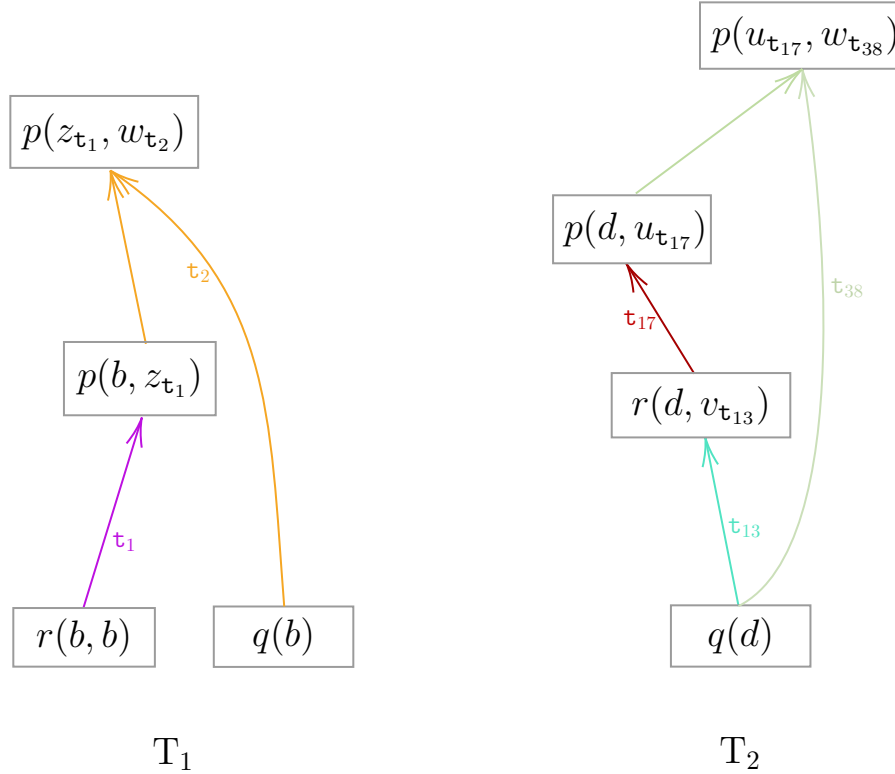
Example 42: Let \mathcal{R} be the following ruleset:

$$R_1 = r(x, y) \rightarrow \exists z p(x, z)$$

$$R_2 = p(x, y) \wedge q(x) \rightarrow \exists w p(y, w)$$

$$R_3 = q(x) \rightarrow \exists v r(x, v)$$

and suppose that $S = \{p(x, y)\}$. Below we find two ancestor trees $T_1 \in \mathbf{T}(S, \mathcal{R}, 2)$ and $T_2 \in \mathbf{T}(S, \mathcal{R}, 3)$:



Notice that the actual variable names do not matter (the renamings according to the definition are $\theta_1 = \{x \mapsto z_{t_1}, y \mapsto w_{t_2}\}$ and $\theta_2 = \{x \mapsto u_{t_{17}}, y \mapsto w_{t_{38}}\}$). ■

The algorithm that searches for ancestor trees is directly based on the following remark:

Remark 5.10. Let S be an atomset, \mathcal{R} a ruleset and $T \in \mathbf{T}(S, \mathcal{R})$. Let S' be a subset of the atoms that are leaves in T and S'' a generator of S with \mathcal{R} . Let T' be the graph obtained by adding S'' and the corresponding edges to T . Then $T' \in \mathbf{T}(S, \mathcal{R})$. ♣

At this point we present an algorithm for rewriting ancestor trees:

ALGORITHM 2: Input: \mathcal{R} , T .

- 1) Choose any set of leaf atoms S of T .
- 2) Find any generator S' of S with \mathcal{R} .
- 3) Add to T all the atoms of S' , with an edge to the atoms that they generate.
- 4) Output T .

Proposition 5.9. For any atomset S , ruleset \mathcal{R} and number k , the set $\mathbf{T}(S, \mathcal{R}, k)$ is computable.

Proof: Lemma 5.1 assures that the set of generators of any atomset is finite up to quasi-isomorphism. So following also the remarks 5.9 and 5.10, by applying Algorithm 2 a finite number of times, we can generate all the ancestor trees of up to a certain depth (k in this case). \square

In the previous section we showed that for all the chase variants that preserve ancestry, it is decidable to determine k -boundedness, by testing only factbases that include no more atoms than the potential ancestors of an atom produced in rank k . These factbases would be generated randomly, but their size is bounded which assures the decidability of the problem. Using the technique presented here we can actually generate factbases with more accurate potential ancestors, because the ancestor trees guarantee that the particular factbases can produce the desired atoms. Then we only need to verify that this can be done in an X-chase scenario (if we are investigating X- k -boundedness).

The computational cost for this procedure in the worst case is higher than the random generation of factbases of a limited size based on a given vocabulary. In particular, with a simple adaptation of the proof of Theorem 5.5 we conclude that the number of quasi-equivalent classes of factbases of a bounded cardinality, is exponential to that cardinality and to the maximal arity of the predicates. On the other hand, the number of generators of a single atom is already exponential to the maximum number of disappearing variables in a rule,

which suggests that rewriting ancestor trees is multiple times exponential with respect to the parameters of a given ruleset (considering Algorithm 2). Nevertheless, the generation of ancestor trees could be useful to improve practical runtime in specific cases, i.e. when the rulesets have convenient characteristics such as linear rules, few or no disappearing variables, etc.

More importantly, ancestor trees have the advantage that they do not depend on the chase variant. Therefore it is interesting to consider them when researching X - k -boundedness in the case that X does not preserve ancestry. From Theorem 5.8 and Corollary 5.6 we know that computing k -minimal X -chase graphs can be a key to determining $(k-)$ X -boundedness. The theorem below shows that when a chase variant preserves ancestry, it is indeed possible to compute all k -minimal X -chase graphs for a certain atomset with a certain ruleset. When a chase variant does not preserve ancestry, it is an open question whether k -minimal chase graphs are computable, and ancestor trees can contribute to answering this question.

Theorem 5.11. Let X be a chase variant that preserves ancestry, S an atomset and \mathcal{R} a ruleset. Then the set $\mathbf{B}_X^k(S, \mathcal{R})$ of all X -chase graphs that are k -minimal for S with \mathcal{R} is computable.

Proof: Since the X -chase preserves ancestry, every graph $G = (V, E)$ in $\mathbf{B}_X^k(S, \mathcal{R})$ will have the property that $V^0 = \text{Anc}_G^0(S)$. But then there is a $T \in \mathbf{T}(S, \mathcal{R}, k)$ such that the leaves of depth k in T comprise exactly V^0 . So by trying to construct X -derivations on V^0 , that retain the rest of the structure of T , we will arrive at G . Thus, in order to generate all the k -minimal X -chase graphs for S with \mathcal{R} , we start from k -deep ancestor trees of S with \mathcal{R} , and we compute X -derivations based on their deeper leaves. If those X -derivations preserve the whole structure of the ancestor tree, then they constitute k -minimal X -chase graphs. \square

6 Conclusion

In this last chapter we provide an overview of our contributions as well as a prospective for some future research. We specifically give an account of a number of open problems that are waiting to be solved.

6.1 Summary

To the best of our knowledge this is the first work that substantiates the notion of *chase variant* in such abstract yet formally concrete terms: we provided a general definition of a derivation and defined a chase variant as a class of derivations. In this way we were able to define properties on chase variants which are helpful in comparing them and getting a deeper understanding of their structure. In Tables 6.1 and 6.2 there is a comprehensive account of the most significant properties (as specified in Definitions 4.17, 5.8, 5.6, 5.7, 5.3 and 5.2 respectively) of chase variants, outlining the established results.

	O	bf-O	SO	bf-SO	R	bf-R	E
Termination-order independence	✓	✓	✓	✓	✗	✗	✓
Depth-order independence	✗	✓	✗	✓	✗	✗	✓
Heredity	✓	✓	✓	✗	✓	✗	✗
bf-R-Compliance	✗	✓	✗	✓	✗	✓	✗
Preservation of Ancestry	✓	✓	✓	✓	✓	✓	✗
Decidability of k -Boundedness	✓	✓	✓	✓	✓	✓	?

Table 6.1: Monotonic chase variants' properties.

	P	LC	F	bf-F	V	bf-V	C
Termination-order independence	✓	✓	✗	✗	✗	✗	✓
Depth-order independence	✓	✓	✗	✗	✗	✗	✓
Heredity	✗	✗	✗	✗	✗	✗	✗
bf-R-Compliance	✓	✓	✗	✗	✗	✗	✗
Preservation of Ancestry	✓	✗	✗	✗	✗	✗	✗
Decidability of k -Boundedness	✓	?	?	?	?	?	?

Table 6.2: Non-monotonic chase variants' properties.

A lot of our work concerns the discerning of characteristics of chase variants and of forward chaining in general which at first sight might appear as details, but eventually they lead to considerably different behaviors. In this way we expose the complexity that lies beneath the simple and intuitive idea of the chase, while we also establish a concrete formal foundation. A prime example for the many nuances that we have shed some light on, is the comparison between **bf-R**-chase, **rc-R**-chase and **P**-chase. The **rc-R**-chase (rank compatible restricted chase) is not included in Table 6.1 but it satisfies the same three of those six properties as the **R**-chase. All the derivations in those three chase variants use triggers of increasing rank that are **R**-applicable to the current (active) factbase. However the seemingly subtle differences in the definitions lead to chase variants with very different properties, as we can see in the above tables.

We used the unique naming of new variables to define the chase space, as a setting that is useful in comparing different derivations from the same knowledge base. We also took advantage of the unique naming of new variables in order to accentuate the role of the triggers in forward chaining. The focus on triggers rather than atoms and rules facilitates a lot of the technical handling of derivations¹. Additionally, we underlined the connection between retraction

¹We note that we can reduce further the formalism if we impose a unique naming of variables in the ruleset (each variable appears in at most one rule), because then the triggers correspond to homomorphisms instead of pairs of homomorphisms and rules. We did not do that because we chose to simplify the presentation of the rulesets rather than the derivations.

and redundancy in the chase, and defined two new chase variants (V-chase & LC-chase) which optimize some known chase variants with respect to elimination of redundancy in the factbase and in rule applications.

Since our research team is the first to extensively work on the notion of boundedness for existential rules, this thesis is the first large-scale publication devoted to that. We approached the problem incrementally, defining k -boundedness for which we also provided a theoretical characterization, utilizing the concept of k -minimal chase graphs. We identified a property for chase variants that ensures the decidability of the problem of determining whether a ruleset is k -bounded. This property is preservation of ancestry and we showed that it happens to also guarantee the decidability of the computation of k -minimal chase graphs. We showed that a number of chase variants preserve ancestry, again by using intermediate properties and theoretical results.

6.2 Future Work

The primary purpose of this research is to aid in the development of knowledge representation systems. Hence an obvious use of our work is to serve as an abstract model for programming forward chaining and developing systems where k -boundedness and other properties of existential rulesets and knowledge bases can be tested. The main trade-off when designing a chase algorithm is between the rapidity of computing derivations and the strength in eliminating redundancy. It would be interesting to compare the chase variants on real-world knowledge bases. How does the LC-chase behave in practice compared to the R-chase and the C-chase with respect to runtime and effective redundancy elimination? This kind of experimental comparison can lead to a better understanding and better solutions for performing robust forward chaining.

On the theoretical side, our research gives rise to a plentiful of new questions. We present a list (with no particular order) of open questions which are not necessarily independent. Below we denote with **rc-X-chase** the class of rank compatible X-derivations and with **lin-X-chase** the chase variant that cor-

responds to the class of X -derivations with a linear ruleset²:

1. Is the problem of X - k -boundedness decidable for the chase variants $X \in \{\mathbf{E}, \mathbf{F}, \mathbf{bf-F}, \mathbf{V}, \mathbf{bf-V}, \mathbf{LC}, \mathbf{C}\}$?
2. Does the \mathbf{LC} -chase loosely preserve ancestry?
(a positive answer would lead to the decidability of \mathbf{LC} -boundedness)
3. *Characterization of X - k -boundedness with k -minimal X -chase graphs:*
Does it hold that X - k -boundedness is decidable if and only if for every atomset S and ruleset \mathcal{R} , the set $\mathbf{B}_X^k(S, \mathcal{R})$ of all X -chase graphs that are k -minimal for S with \mathcal{R} is computable?
4. *Variation of the Stable Rank Theorem for \mathbf{C} -chase:* Let $\mathbb{C}(F, \mathcal{R})$ be a chase space and let \mathcal{D} be a \mathbf{C} -derivation from (F, \mathcal{R}) . Does it hold that for every $A \in F^{\mathcal{D}}$ we have $\text{rank}_{\mathcal{D}}(A) = \text{rank}_{(F, \mathcal{R})}(A)$?
5. When $X \in \{\mathbf{R}, \mathbf{F}, \mathbf{V}\}$, is the $\mathbf{rc-X}$ -chase equivalent with the X -chase with respect to termination?
6. Is the all-factbase $\mathbf{lin-R}$ -chase termination decidable?
7. Does the $\mathbf{lin-C}$ -chase (loosely) preserve ancestry?

Question 7 seems considerably simplified if we consider the sub-variant that does not include any variables in the initial factbases. Questions 6 and 7 are motivated by Proposition 5.2: indeed, according to this proposition, the decidability of all-factbase $\mathbf{lin-R}$ -chase termination decidable is equivalent the decidability of $\mathbf{lin-R}$ -boundedness³. Furthermore, we know that the all-factbase $\mathbf{lin-C}$ -chase termination is decidable [21]. So if the $\mathbf{lin-C}$ -chase preserves ancestry, then we can use Proposition 5.2 to conclude that $\mathbf{lin-C}$ -boundedness

²This is a substantiation of the versatility of the notion of chase variant as introduced in this thesis, as it shows that syntactic restrictions imposed in the knowledge base can also be represented in chase variants, hence all properties defined for chase variants can be respectively tested for particular rule classes (datalog, linear, acyclic, guarded, etc).

³In fact as mentioned in earlier chapters, it has been shown [22] that \mathbf{R} -chase termination is decidable for extra linear rulesets, whose only difference from linear rulesets is in the size of the rule head (where we have only one atom). And while for the \mathbf{O} - and \mathbf{SO} -chase, decomposing rule heads does not change any properties with respect to boundedness and termination, for the \mathbf{R} -chase it does.

is decidable. Moreover the proof of this proposition can be trivially modified so as to arrive to the same conclusion for chase variants that only loosely preserve ancestry. So even if the **lin-C**-chase does not preserve ancestry, if we prove that it loosely preserves ancestry then we will still have shown that **lin-C**-boundedness is decidable.


Those seven questions stem directly from the material presented in Chapter 4 and Chapter 5. However there are several less direct but quite intuitive extensions to our definitions that merit mentioning and that open another Pandora's box of new puzzles to be solved.

The most important is a weaker form of boundedness:

Definition 6.1. Let X be a chase variant. A ruleset \mathcal{R} is \exists - X -*bounded* if there is a $k \in \mathbb{N}$ such that for every factbase F , there exists a terminating X -derivation from (F, \mathcal{R}) of depth at most k . Respectively, we say that \mathcal{R} is \exists - X - k -*bounded* if for every factbase F , there exists a terminating X -derivation from (F, \mathcal{R}) of depth at most k . \dashv

The motivation for the above definition is that given a knowledge base, our primary interest is to find a finite universal model. And for all the chase variants that we have presented, terminating derivations produce universal models. Hence even if not all exhaustive X -derivations are terminating, it is useful to know if there is some way that the X -chase will terminate. In a broader sense, given a ruleset, it is useful to know if for every factbase there always exists a terminating X -derivation of a bounded depth.

In the case depth-order independent chase variants, it is clear that the two notions of boundedness coincide:

Remark 6.1. Let X be a depth-order independent chase variant. Then a ruleset \mathcal{R} is X -bounded if and only if it is \exists - X -bounded. 

So in a trivial way we conclude that \exists - X - k -boundedness is decidable when $X \in \{\mathbf{bf-O}, \mathbf{bf-SO}, \mathbf{P}\}$. Moreover from Proposition 4.6 we find that for $X \in \{\mathbf{O}, \mathbf{SO}\}$ it holds that \exists - X - k -boundedness is equivalent with \exists -**bf**- X - k -boundedness. Thus \exists - X - k -boundedness is also decidable when

$X \in \{\mathbf{O}, \mathbf{SO}\}$. Notice that the \mathbf{O} -chase and \mathbf{SO} -chase are the only chase variants that we have seen that are depth-order sensitive but termination-order independent (see Table 6.1).

On the other hand the study of the behavior of termination-order dependent chase variants with respect to \exists -X-boundedness appears interesting and challenging. Concepts like preservation of ancestry do not appear to lead to solutions and it seems that a whole different approach might be needed to solve this problem, which comprises the 8th of our open questions:

8. Is the problem of \exists -X- k -boundedness decidable for the chase variants $X \in \{\mathbf{R}, \mathbf{bf-R}, \mathbf{F}, \mathbf{bf-F}, \mathbf{V}, \mathbf{bf-V}\}$?

We turn now to the discussion concerning elimination of redundancy, as introduced in Section 4.4. We mentioned that two chase variants can be independent with respect to eliminating redundancy and based on a previous example, we showed that the \mathbf{F} -chase and the \mathbf{E} -chase are independent with respect to elimination of redundancy. Similarly it holds that the \mathbf{C} -chase and the \mathbf{E} -chase are independent with respect to the elimination of redundancy: the knowledge base of the same example (Example 13) can be used to show that the \mathbf{E} -chase is not as strong as the \mathbf{C} -chase in eliminating redundancy. To show the other direction we can use the knowledge base $(\{p(a), p(b)\}, \{p(x) \rightarrow \exists z r(z)\})$: the \mathbf{E} -chase will terminate after one rule application whereas the \mathbf{C} -chase will terminate after two rule applications. In our following 9th open question, by chase variant we will mean *universal chase variant*: a chase variant such that every terminating derivation produces a universal model of the knowledge base.

9. Is there a chase variant which is stronger (in eliminating redundancy) than the \mathbf{E} -chase? Is there a chase variant which is stronger than the \mathbf{C} -chase? Is there a chase variant that is stronger than any other chase variant?

Our intuition says that if there is a universal chase variant that is stronger than any other universal chase variant, then this will not be a breadth-first chase variant. It will therefore be interesting to see what kind of trigger prioritization will be used in order to define this chase variant.

Our final question will be less strictly formalized. We start by noting that the conjunctive query rewriting procedure can be designed in a such a way that it terminates within k breadth-first steps, where k is the bound for the core chase [38]. This implies that if we can define a concept of *depth* of query rewriting such that a ruleset \mathcal{R} will be \mathbf{C} - k -bounded if and only if the number k is a bound to the depth of query rewriting with \mathcal{R} . Hence \mathbf{C} -boundedness (which is also equivalent to \mathbf{E} -boundedness) characterizes the query rewriting potential of a ruleset.

10. For every chase variant \mathbf{X} , translate \mathbf{X} -boundedness to a query rewriting feature. In other words find a property of query rewriting with a ruleset \mathcal{R} which is equivalent with the \mathbf{X} -boundedness of \mathcal{R} .

This concludes our list of future prospective and open problems.

APPENDIX

(A) Query Rewriting in Existential Rules

At subsection 2.1.3 we introduced the notion of piece in an atomset. This notion can be extended to heads of rules, by using existential variables as the connecting terms. We present a piece-based query rewriting mechanism which is sound and complete for existential rules, and terminating for fus rulesets [44, 1].

Definition .2 (Piece). Let $R = (B, H)$ be an existential rule. Every atom $A \in H$ is *connected* with itself. Two atoms $A, A' \in H$ are *connected* (by existential variables) if there is a sequence $A_1, \dots, A_n \in H$ such that $A_1 = A$, $A_n = A'$ and for every $i < n$ holds that there exists a $z \in \mathbf{exv}(R) \cap \mathbf{var}(A_i) \cap \mathbf{var}(A_{i+1})$. A *piece* in the head of R is a maximal (non empty) set of connected atoms. \dashv

The main difference between the notion of piece in a factbase (as specified in Definition 2.5) and the notion of piece in the head of the rule as defined above, is that in the latter case the atoms are connected by existential variables whereas in the former case they are connected by any variables. The notion of piece in the head of a rule is necessary in order to define query rewriting with existential rules. The following example motivates the involved definition of piece unifier which follows.

Example “QUERE” 43: Take the rule $R = p(x) \rightarrow \exists z q(x, z)$ and let $Q_0 = \{q(x_0, x_1), r(x_0)\}$ and $Q_1 = \{q(x_0, x_1), r(x_1)\}$. We want to know if the queries Q_0 and Q_1 can be included in a factbase produced by an application of the rule R to some factbase. It is easy to verify that R can produce an atom

isomorphic to $q(x_0, x_1)$. But then x_1 is a new variable, introduced with this rule application. Hence it cannot appear in any other atoms of the resulting factbase. Therefore we know that $q(x_0, x_1)$ in Q_1 cannot result from an application of R . So we cannot use R to rewrite Q_1 . On the other hand, Q_0 can be produced by an application of R . In particular if $F = \{p(x_0), r(x_0)\}$, then one application of R to F produces a factbase that includes Q_0 . ■

Let T be a set of terms and $P = \{T_1, T_2, \dots\}$ a partition of T . We call P *admissible* if all its elements contain at most one constant. Let $f : P \rightarrow T$ be an injection such that for all i holds $f(T_i) \in T_i$ and if there is a constant $c \in T_i$ then $f(T_i) = c$. A substitution $\sigma : \text{var}(T) \rightarrow T$ is *associated* with P if there exists an injection f as specified above such that for every i , if $x \in T_i$ then $\sigma(x) = f(T_i)$.

Definition .3 (Piece-Unifier[1]). Let $R = (B, H)$ be an existential rule and Q be a query. Let $\emptyset \neq Q' \subseteq Q$, $H' \subseteq H$ and $T = \text{term}(Q' \cup H')$. A substitution $\mu : \text{var}(Q' \cup H') \rightarrow T$ associated with an admissible partition of T is a *piece-unifier* of Q with R with respect to (Q', H') , if $\mu(Q') = \mu(H')$ and

- for all $z \in \text{exv}(R)$, $\mu(z)$ is not a constant,
- for all $z \in \text{exv}(R)$ and $x \in \text{var}(Q' \cup H')$ with $z \neq x$, if $\mu(z) = \mu(x)$ then $x \in (\text{var}(Q) \setminus \text{var}(Q' \cup H'))$.

The set of all the piece-unifiers of a query Q with a rule R is symbolized with $\text{pun}(Q, R)$. ⊥

The last condition ensures that each existential variable can be unified only with variables that appear exclusively in Q' (and not elsewhere in Q). If μ is a piece unifier wrt (Q', H') we will say that μ *unifies* Q' with H' . Note that the above definition can trivially be expanded to unify the body of a rule with the head of another rule, in order to introduce the notion of *composition of rules* (which is beyond the scope of this thesis).

Now using the notion of piece-unifier we can formally define what is a *rewriting* of a query with an existential rule:

Definition .4. Given a query Q , a rule R and a piece-unifier μ of Q with R wrt (Q', H') , the *immediate rewriting* of Q according to μ , denoted $\beta(Q, R, \mu)$ is $\mu(\text{body}(R)) \cup \mu(Q \setminus Q')$. \dashv

Example “QUERE” 44 (continued from Example 43): Using R and Q_0 as specified above, we have that $\mu = \{x \mapsto x_0, z \mapsto x_1\}$ is a piece unifier of Q_0 with R with respect to $(\{q(x_0, x_1)\}, \{q(x, z)\})$. So we can use R to rewrite Q_0 , resulting in the rewriting $\beta(Q_0, R, \mu) = \{p(x_0), r(x_0)\}$. \blacksquare

We can expand the notion of immediate rewriting to a breadth-first process where at each level we produce all the possible rewritings. To that end, given a ruleset \mathcal{R} and a UCQ \mathcal{Q} (which we always consider as a set of boolean conjunctive queries), we can define a “naive” operator β with

$$\beta(\mathcal{Q}, \mathcal{R}) = \mathcal{Q} \cup \{\beta(Q, R, \mu) \mid Q \in \mathcal{Q}, R \in \mathcal{R}, \mu \in \text{pun}(Q, R)\}$$

and we denote the repeated applications of β with an index: $\beta_0(\mathcal{Q}, \mathcal{R}) = \mathcal{Q}$ and $\beta_{i+1}(\mathcal{Q}, \mathcal{R}) = \beta(\beta_i(\mathcal{Q}, \mathcal{R}), \mathcal{R})$ for all $i \geq 0$. Finally we have $\beta_\infty(\mathcal{Q}, \mathcal{R}) := \bigcup_{i \in \mathbb{N}} \beta_i(\mathcal{Q}, \mathcal{R})$. This operator is sound and complete: $F \cup \mathcal{R} \models Q$ if and only if there exists $Q' \in \beta_\infty(Q, \mathcal{R})$ such that $F \models Q'$ (or equivalently, if there is a $k \in \mathbb{N}$ such that there is a $Q' \in \beta_k(Q, \mathcal{R})$ with $F \models Q'$). It also has the following convenient property:

Proposition .1 ([1, 41, 8]). A ruleset \mathcal{R} is fus if and only if for every query Q there is a $k \in \mathbb{N}$ such that $\beta_\infty(Q, \mathcal{R}) \equiv \beta_k(Q, \mathcal{R})$. \clubsuit

(B) Properties of the Semi-Oblivious Chase

In this section we show that the semi-oblivious chase is closely related to the oblivious chase in aspects pertaining to termination and depth. By choosing a different name for the new existential variables, we arrive at a simple and intuitive transformation of the notion of derivation as defined in this thesis, to a notion of *meta-derivation* which produces isomorphic results, while making no distinction between SO-chase and O-chase. But before specifying this transformation, we provide a detailed proof for the SO-case of Proposition 4.6. This proof serves as an exercise in order to identify the problem that relates with the naming of existential variables but it is redundant considering the transformation that we present immediately afterwards:

Proposition .2. For each terminating SO-derivation from (F, \mathcal{R}) there exists a breadth-first terminating SO-derivation from (F, \mathcal{R}) of smaller or equal depth.

Proof: **Case SO:** Let \mathcal{D} be a terminating SO-derivation from (F, \mathcal{R}) . As per Remark 4.3.iii, when O-applicability is secured, the condition for SO-applicability is non-SO-equivalence. Hence, we can rearrange $\mathbf{trig}(\mathcal{D})$ in a rank compatible manner, obtaining \mathcal{T} , and then we have a SO-derivation \mathcal{D}' from (F, \mathcal{R}) with $\mathbf{trig}(\mathcal{D}') = \mathcal{T}$. So \mathcal{D}' is rank compatible and moreover it is terminating (otherwise \mathcal{D} would also not be terminating). And from Proposition 4.5, we obtain that \mathcal{D}' is of smaller or equal depth than \mathcal{D} . However we do not know if \mathcal{D}' is breadth-first. Nevertheless we can transform it to a breadth-first derivation. Let $m = \max\{\mathit{rank}(\mathbf{t}) \mid \mathbf{t} \in \mathbf{trig}(\mathcal{D}')\}$. In what follows we will use the notation \mathcal{D}'^k to represent the prefix of \mathcal{D}' that includes all elements of rank up to k . We define the following algorithm:

ALGORITHM 3: Input: $(F, \mathcal{R}), \mathcal{D}'$

1) **for** $k = 1$ to m ,

I) **while** there is a trigger $\mathfrak{t} \notin \text{trig}(\mathcal{D}')$ that is **SO**-applicable on a prefix \mathcal{D}'^k of \mathcal{D}' ,

i) find $\mathfrak{t}' \in \text{trig}(\mathcal{D}')$ which is **SO**-equivalent with \mathfrak{t} , let $\tau : \text{nul}(\text{op}(\mathfrak{t}')) \rightarrow \text{nul}(\text{op}(\mathfrak{t}))$ be the isomorphism with $\tau(\text{op}(\mathfrak{t}')) = \text{op}(\mathfrak{t})$.

ii) declare a sequence of triggers $\bar{\mathcal{T}}$ and set $\bar{\mathcal{T}} = \text{trig}(\mathcal{D}'^k)$,

iii) add \mathfrak{t} to $\bar{\mathcal{T}}$,

iv) add to $\bar{\mathcal{T}}$ all the triggers $\mathfrak{t}'' \in \text{trig}(\mathcal{D}') \setminus \{\mathfrak{t}'\}$ with the property $k + 1 \leq \text{rank}(\mathfrak{t}'') \leq \text{rank}(\mathfrak{t}')$,

v) **for** $i = \text{rank}(\mathfrak{t}') + 1$ to m ,

- declare a set of mappings (substitution) $\bar{\tau}$, initially empty.

- for all triggers $\mathfrak{t}^i = (R, \pi) \in \text{trig}(\mathcal{D}')$ with $\text{rank}(\mathfrak{t}^i) = i$,

· set $\mathfrak{f}^i := (R, \tau \circ \pi)$,

· add \mathfrak{f}^i to $\bar{\mathcal{T}}$ and

· for every $x \in \text{nul}(\text{op}(\mathfrak{t}^i))$ add $\{x_{\mathfrak{t}^i} \mapsto x_{\mathfrak{f}^i}\}$ to $\bar{\tau}$
(if $\mathfrak{t}^i = \mathfrak{f}^i$ then the mapping is the identity).

- Set $\tau := \bar{\tau} \circ \tau$.

vi) reorder $\bar{\mathcal{T}}$ according to rank.

vii) \mathcal{D}'' is the rank compatible **SO**-derivation from (F, \mathcal{R}) with $\text{trig}(\mathcal{D}'') = \bar{\mathcal{T}}$.

viii) set $\mathcal{D}' := \mathcal{D}''$, go back to step I).

2) Output \mathcal{D}' .

In every iteration of the loop which appears in step 1 of Algorithm 3, we detect one trigger \mathfrak{t} that is **SO**-applicable at some intermediate rank k in \mathcal{D}' . From Remark 4.3.iii we know that there is a trigger \mathfrak{t}' in $\text{trig}(\mathcal{D}')$ that is **SO**-equivalent with \mathfrak{t} . And since \mathfrak{t}' does not appear in \mathcal{D}'^k we know that $\text{rank}(\mathfrak{t}') > k$. What we want to do, is to replace \mathfrak{t}' with \mathfrak{t} in \mathcal{D}' . After all, their outputs are isomorphic. There is nonetheless a technical detail which we have to take care of: there are other triggers in \mathcal{D}' which depend on \mathfrak{t}' , i.e. their support includes at least an atom from the output of \mathfrak{t}' . The rest of the algorithm is dedicated in

performing all the renamings coherently so that the swapping of τ' with τ does not change the overall interdependence of triggers and overall functionality of \mathcal{D}' . This is achieved by constructing a new sequence $\bar{\mathcal{T}}$ of triggers which will specify the altered derivation.

At first we note that the prefix \mathcal{D}'^k need not change, so we start with $\bar{\mathcal{T}} = \text{trig}(\mathcal{D}'^k)$. We apply τ on \mathcal{D}'^k and then we can apply all triggers that are of rank at most $\text{rank}(\tau')$, since they do not depend on τ' . This explains steps 1.I.iii and 1.I.iv of the algorithm. Let $i = \text{rank}(\tau') + 1$. At rank i , we want to “reorient” the supports of all triggers, from the new variables of $\text{op}(\tau')$ to the new variables of $\text{op}(\tau)$. Since there is an isomorphism τ between the two atomsets, we simply need to compose this isomorphism with each homomorphism π in every trigger of rank more than $\text{rank}(\tau')$. This happens by transforming every τ^i to τ^i in step 1.I.v. of Algorithm 3. In this way the same rules will be applied, only this time to the (isomorphic) atoms produced by τ instead of τ' . The problem that occurs is that then at the following rank ($i + 1$), we will need to reorient not only the triggers that depend on τ' in \mathcal{D}' , but also those that depend on the triggers of rank i which we have altered. This is why we need to keep track of all the triggers τ^i introduced in rank i and in particular we declare the set of mappings $\bar{\tau}$, where the new variables of the old triggers are mapped to the new variables of the reoriented triggers. Then, in order to work on the next rank, we need to compose τ with $\bar{\tau}$. This kind of repairing continues until the highest rank m .

Finally at step 1.I.vi we reorder $\bar{\mathcal{T}}$ according to rank. In this way we obtain a rank compatible SO-derivation \mathcal{D}'' where τ is replaced by τ' and subsequent triggers are “reoriented” accordingly (i.e. swapped with SO-equivalent triggers applicable to the changed new variables). \mathcal{D}'' is still terminating because for every trigger that we removed, we added one that is SO-equivalent, which guarantees also that $F^{\mathcal{D}'}$ is isomorphic to $F^{\mathcal{D}''}$. Finally, to show that the depth of \mathcal{D}'' is smaller or equal to that of \mathcal{D}' , note that $\text{rank}_{\mathcal{D}''}(\tau) < \text{rank}_{\mathcal{D}'}(\tau')$ implies that for every atom A produced by τ in \mathcal{D}'' holds that $\text{rank}_{\mathcal{D}''}(A) < \text{rank}_{\mathcal{D}'}(\tau^{-1}(A))$. And the same holds for all atoms produced by the reoriented triggers τ^i of step 1.I.v. Furthermore, the rank compatible trigger sorting of step 1.I.vi does not

augment the depth, as per Proposition 4.5.

After that we set $\mathcal{D}' = \mathcal{D}''$ and repeat until there are no more triggers SO-applicable on \mathcal{D}'^k . To see that the while loop terminates, note that the set $\text{trig}(\mathcal{D}') \setminus \text{trig}(\mathcal{D}'^k)$ is smaller after each iteration. Lastly, when the algorithm terminates we know that the final \mathcal{D}' will be breadth-first because at every rank there are no more SO-applicable triggers. And the final \mathcal{D}' is of smaller or equal depth than the original \mathcal{D}' since this property is retained with every iteration of the while loop. \square

In the above algorithm, notice the hassle that comes with redefining the triggers in the extension of the SO-derivation each time we replace a trigger with a SO-equivalent trigger in a lower rank. A way to avoid all this is to declare a different kind of naming of the new variables. In particular, let \mathbf{f} represent a SO-equivalence class of triggers. We call \mathbf{f} a *meta-trigger* and we can understand it as a trigger (R, π) whose substitution π does not specify a mapping for the disappearing variables of the respective rule R . A meta-trigger \mathbf{f} is *applicable* on a factbase F if there is any trigger $\mathbf{t} \in \mathbf{f}$ that is applicable on F . Then every existential variable z in the head of R can be named $z_{\mathbf{f}}$, defining the output of the \mathbf{f} , which we denote then with $\text{op}(\mathbf{f})$. Using meta-triggers instead of triggers we construct *meta-derivations* (as in Definition 4.1). The notions of *rank* and *depth* can then be defined for meta-triggers (and the atoms they produce in the meta-derivation) as the lowest ranks among the triggers they include.

Proposition .3. Every SO-derivation can be translated into a meta-derivation producing isomorphic atoms at the same ranks. Conversely, every meta-derivation can be specialized into a SO-derivation producing isomorphic atoms at the same ranks.

Proof: We can replace each trigger of rank 1 with a meta-trigger and then we follow the recursive tactic as employed in Algorithm 3, to rename all the new variables and the supported triggers rank by rank. The inverse is also possible by specifying mappings for the disappearing variables. \square

We can define a new chase space based on meta-derivations. This can cause

up to several SO-equivalence classes of triggers to collapse to a single meta-trigger (since the common variable name will unite distinct SO-equivalence classes of higher ranks). With this transformation we effectively reduce the chase space in such a way that the SO-chase (represented by the class of all meta-derivations) behaves exactly as the O-chase does in the original chase space. Notice specifically that the application of a meta-trigger on a meta-derivation does not cancel the applicability of any other meta-trigger on the resulting extension. This leads us to conclude that:

Remark .2. Every exhaustive SO-chase derivation from (F, \mathcal{R}) necessarily includes all meta-triggers f that can appear in any meta-derivation from (F, \mathcal{R}) .

As a result we know that an exhaustive meta-derivation from a knowledge base (F, \mathcal{R}) is terminating if and only if the set of all meta-triggers on (F, \mathcal{R}) is finite. Hence:

Corollary .1. The SO-chase is termination-order independent. ♣

Finally we show the following:

Proposition .4. The bf-SO-chase is depth-order independent.

Proof: Since the application of meta-triggers does not cancel the applicability of other meta-triggers, we know that all meta triggers applicable to the initial factbase will be applied at rank 1. Hence every breadth-first meta-derivation of rank 1 from the same knowledge base will infer exactly the same atoms and include exactly the same triggers. Similarly we reach the same conclusion for every rank i . Therefore all exhaustive breadth-first meta-derivations will be of the same depth. □

(C) Partial Core & Freezing

In this section we connect the notion of partial core (specified in Section 5.4, Definition 5.9) with the computation of a core of an atomset. This is possible by *freezing* the preserved variables, i.e. converting them to constants, and then calculating a core. But to be consistent we need to re-convert the newly introduced constants to variables after this calculation. To this end we assume that every variable x in our vocabulary uniquely corresponds to a *dedicated constant* a_x . Here is how these concepts can be more formally presented:

Definition .5 (Partial Freezing). Let F be an atomset and $W \subseteq \text{var}(F)$. The W -freezing (denoted \wp) is a substitution that maps every variable $x \in W$ to a dedicated constant a_x . So we write $\wp(F)$ to represent the image of the W -freezing on F (also called W -freezing of F). The W -unfreezing \wp^* is the inverse mapping, from each dedicated constant to its respective variable (and we write $\wp^*(F)$). \dashv

For ease of presentation, when applied on a single symbol, we will drop the first parentheses for the freezing and unfreezing operators, writing $\wp F$ and $\wp^* F$ instead of $\wp(F)$ and $\wp^*(F)$ respectively. Note that an unfreezing is not a substitution but an injective mapping from constants to variables.

Remark .3 (Properties of Partial Freezing). For every atomset F , retraction h on F and variable sets W and U it holds that:

- i. if $W = \emptyset$ we get that $\wp F = F$.
- ii. $\wp(\wp^* F) = \wp^*(\wp F)$.
- iii. if $V = W \cup U$ then $\wp F = \wp(\wp^* F)$.
- iv. if $\text{dom}(h) \cap W = \emptyset$ then the substitution $\wp h = \wp \circ h$ is a retraction on $\wp F$ and it holds that $\wp h(\wp F) = \wp(h(F))$. \clubsuit

The core is a minimal retract. Similarly, the partial core is a minimal retract preserving certain variables. Based on the above notion of partial freezing, we can give another characterization of the concept of partial core:

Proposition .5. Let F be an atomset and $W \subseteq \text{var}(F)$. A partial core of F preserving W is the unfreezing of a core of the W -freezing of F , i.e. $\text{pcore}(F, W) = \mathbb{W}(\text{core}(\mathbb{W}F))$. ♣

Just as is the case with the core, the partial core is non-deterministic as an operation on an atomset. In other words, there can be many partial cores of an atomset preserving a particular variable-set. Hence by specifying $\text{pcore}(F, W) = F'$, we are choosing any of the partial cores of F preserving W , and designating it as F' . But if F' is already defined, it is more accurate to say that it *belongs to the set of partial cores* of F preserving W . We use bold notation to denote this set, i.e. $\text{pcores}(F, W)$, so the statement can be written as $F' \in \text{pcores}(F, W)$. As already outlined in Chapter 2, we are also using this kind of notation for the notion of core, so $\text{core}(F)$ denotes any core of F while $\text{cores}(F)$ denotes the set of all atomsets that are cores of F .

The following remark underlines the direct connection between cores and partial cores:

Remark .4. Let $F, W \subseteq \text{var}(F)$ and $F' \subseteq F$. Then it holds that $F' \in \text{pcores}(F, W)$ if and only if $\mathbb{W}F' \in \text{cores}(\mathbb{W}F)$. ♣

To further establish the notion of partial core, we prove the following property:

Lemma .1. Let F be an atomset and $W, U \subseteq \text{var}(F)$. Let $F' \subseteq F$. Then $F' \in \text{pcores}(F, W \cup U)$ if and only if $\mathbb{W}F' \in \text{pcores}(\mathbb{W}F, U)$.

Proof: In this proof we are basically transforming formulas based on the remarks .3 and .4.

(\Rightarrow ;) Let $F' \in \text{pcores}(F, W \cup U)$. Let $V = W \cup U$. So $F' \in \text{pcores}(F, V)$. Hence $\mathbb{V}F' \in \text{cores}(\mathbb{V}F)$, which implies $\mathbb{V}(\mathbb{W}F') \in \text{cores}(\mathbb{V}(\mathbb{W}F))$. So by definition 5.9, $\mathbb{U}(\mathbb{V}(\mathbb{W}F'))$ is a partial core of $\mathbb{W}F$ with respect to U . And of course $\mathbb{U}(\mathbb{V}(\mathbb{W}F')) = \mathbb{W}F'$.

(\Leftarrow ;) Let $\mathbb{W}F' \in \text{pcores}(\mathbb{W}F, U)$. Therefore $\mathbb{U}(\mathbb{W}F') \in \text{cores}(\mathbb{U}(\mathbb{W}F))$. Let $F' \in \text{pcores}(F, W \cup U)$. So we have $\mathbb{V}F = \mathbb{W}(\mathbb{V}F)$, thus we can write $\mathbb{V}F' \in \text{cores}(\mathbb{V}F)$. Then by definition $\mathbb{V}(\mathbb{V}F') \in \text{pcores}(F, V)$, so $F' \in \text{pcores}(F, V)$ which is $F' \in \text{pcores}(F, W \cup U)$. □

(D) Frugal Chase

We first remind our definition of frugal chase, from Chapter 4.

An *output piece* of a trigger \mathfrak{t} is a minimal subset $\overline{\mathbf{op}}(\mathfrak{t}) \subseteq \mathbf{op}(\mathfrak{t})$ with the property that if $A \in \overline{\mathbf{op}}(\mathfrak{t})$, then for every $A' \in \mathbf{op}(\mathfrak{t})$, if the atoms A and A' have at least one common new variable, then it holds that $A' \in \overline{\mathbf{op}}(\mathfrak{t})$. Now suppose that $\{\mathbf{op}_1(\mathfrak{t}), \dots, \mathbf{op}_n(\mathfrak{t})\}$ are the output pieces of a trigger \mathfrak{t} and F is a factbase. The *frugal output* $\mathbf{frop}(\mathfrak{t}, F)$ of \mathfrak{t} with respect to F is the union of all the pieces $\mathbf{op}_i(\mathfrak{t}) \subseteq \mathbf{op}(\mathfrak{t})$ which have the property that F is not a retract of $F \cup \mathbf{op}_i(\mathfrak{t})$. Lastly, let $Z \subseteq F^{\mathcal{D}}$, where \mathcal{D} is a derivation. A piece P in Z is *isomorphically subsumed* by $\mathbf{frop}(\mathfrak{t}, Z)$ if there is an isomorphism σ from P to a subset of $\mathbf{frop}(\mathfrak{t}, Z)$ such that σ is also a retraction from $\mathbf{frop}(\mathfrak{t}, Z) \cup P$ to $\mathbf{frop}(\mathfrak{t}, F)$. Then we denote with $\mathbf{F}(\mathfrak{t}, Z)$ the union of all the pieces in Z that are not isomorphically subsumed by $\mathbf{frop}(\mathfrak{t}, Z)$.

Definition .6. Let $\mathcal{D} = (\mathfrak{t}_*, F_*, Z_*)$ be a derivation from (F, \mathcal{R}) . \mathcal{D} is a *frugal derivation* if for all $i > 0$ holds that

- the trigger \mathfrak{t}_i is **R**-applicable to Z_{i-1} and
- $Z_i = \mathbf{F}(\mathfrak{t}_i, Z_{i-1}) \cup \mathbf{frop}(\mathfrak{t}_i, Z_{i-1})$. ⊣

In the original specification of the frugal chase [28, 59], forward chaining is carried out in a monotonic fashion, so no atoms can be removed from the factbase. However, what happens is that the subset(s) of $\mathbf{frop}(\mathfrak{t}, Z)$ that is (are) isomorphic with a piece P in Z , is (are) ignored, and we add the rest of $\mathbf{frop}(\mathfrak{t}, Z)$ by applying the inverse renaming(s), i.e. if σ is an isomorphism from a (maximal) union of pieces of Z to a subset S of $\mathbf{frop}(\mathfrak{t}, Z)$, we add $\sigma^{-1}(\mathbf{frop}(\mathfrak{t}, Z) \setminus S)$ to the factbase Z . This inverse renaming is only possible if we consider isomorphisms (i.e. bijective renamings), which explains why the frugal chase was not defined for any kind of retraction. This leads to an isomorphic result with the one we get using our definition of the frugal chase. But the naming of the new variables is considerably changing, possibly producing atoms that do not belong to the respective chase space, i.e. they cannot be produced by any

derivation (using Definition 4.1). Hence, although the original definition is not syntactically compatible with our (derivation) framework, it is nevertheless semantically compatible, producing isomorphic results.

Bibliography

- [1] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On Rules with Existential Variables: Walking the Decidability Line. *Artificial Intelligence*, 175(9-10):1620–1654, 2011.
- [2] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14:57 – 83, 2012. Special Issue on Dealing with the Messiness of the Web of Data.
- [3] Herve Gallaire and Jack Minker, editors. *Logic and Data Bases*. Perseus Publishing, 1978.
- [4] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-lite: Tractable description logics for ontologies. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 602–607, 2005.
- [5] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 364–369, 2005.
- [6] Serge Abiteboul, Richard Hull, and Victor Vianu, editors. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

- [7] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. Datalog extensions for tractable query answering over ontologies. In *Semantic Web Information Management - A Model-Based Perspective*, pages 249–279. 2009.
- [8] Marie-Laure Mugnier and Michaël Thomazo. An introduction to ontology-based query answering with existential rules. In *Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, pages 245–278, 2014.
- [9] Michaël Thomazo. *Conjunctive Query Answering Under Existential Rules - Decidability, Complexity, and Algorithms*. PhD thesis, Montpellier 2 University, France, 2013.
- [10] Sebastian Rudolph. The two views on ontological query answering. In *Proceedings of the 8th Alberto Mendelzon Workshop on Foundations of Data Management, Cartagena de Indias, Colombia, June 4-6, 2014.*, 2014.
- [11] C. Beeri and M. Y. Vardi. *The implication problem for data dependencies*, pages 73–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.
- [12] Alfred V. Aho, Catriel Beeri, and Jeffrey D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, 1979.
- [13] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.
- [14] Gösta Grahne and Adrian Onet. Anatomy of the chase. *CoRR*, abs/1303.6682, 2013.
- [15] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. 2013.

- [16] B. Marnette. Generalized schema-mappings: from termination to tractability. In *PODS*, pages 13–22, 2009.
- [17] 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.
- [18] Alin Deutsch, Alan Nash, and Jeff Remmel. The chase revisited. In *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS ’08, pages 149–158, New York, NY, USA, 2008. ACM.
- [19] Tomasz Gogacz and Jerzy Marcinkowski. All-instances termination of chase is undecidable. In *ICALP 2014 Proceedings, Part II*, pages 293–304, 2014.
- [20] Marco Calautti, Georg Gottlob, and Andreas Pieris. Chase termination for guarded existential rules. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 91–103. ACM, 2015.
- [21] André Hernich. Computing universal models under guarded tgds. In *15th International Conference on Database Theory, ICDT ’12, Berlin, Germany, March 26-29, 2012*, pages 222–235, 2012.
- [22] Michel Leclère, Marie-Laure Mugnier, Michaël Thomazo, and Federico Ulliana. A single approach to decide chase termination on linear existential rules. In *Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27th - to - 29th, 2018.*, 2018.
- [23] Tomasz Gogacz, Jerzy Marcinkowski, and Andreas Pieris. All-instances restricted chase termination: The guarded case. *CoRR*, abs/1901.03897, 2019.

- [24] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
- [25] Haim Gaifman, Harry Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, July 1993.
- [26] Ke Wang. Some positive results for boundedness of multiple recursive rules. In *Database Theory - ICDT'95, 5th International Conference, Prague, Czech Republic, January 11-13, 1995, Proceedings*, pages 383–396, 1995.
- [27] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.
- [28] George Konstantinidis and José Luis Ambite. Optimizing the chase: Scalable data integration under constraints. *Proc. VLDB Endow.*, 7(14):1869–1880, October 2014.
- [29] Serge Abiteboul. Boundedness is undecidable for datalog programs with a single recursive rule. *Inf. Process. Lett.*, 32(6):281–287, 1989.
- [30] Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. Undecidable boundedness problems for datalog programs. *J. Log. Program.*, 25(2):163–190, 1995.
- [31] Stavros Cosmadakis, Haim Gaifman, Paris Kanellakis, and Moshe Vardi. Decidable optimization problems for database logic programs. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 477–490, New York, NY, USA, 1988. ACM.
- [32] Herbert B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

- [33] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: Getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, March 2005.
- [34] Pavol Hell and Jaroslav Nesetril. *Graphs and Homomorphisms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford, 2004.
- [35] Pavol Hell and Jaroslav Nesetril. The core of a graph. *Discrete Mathematics*, 109(1-3):117–126, 1992.
- [36] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Database Theory - ICDT 2003, 9th International Conference, Siena, Italy, January 8-10, 2003, Proceedings*, pages 207–224, 2003.
- [37] Adrian Onet. The chase procedure and its applications in data exchange. In *Data Exchange, Integration, and Streams*, pages 1–37. 2013.
- [38] Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. On Bounded Positive Existential Rules. In *29th International Workshop on Description Logics*, Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 2016. [ceur-ws](#).
- [39] 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. *CoRR*, abs/1406.4110, 2014.
- [40] David Maier, Jeffrey D. Ullman, and Moshe Y. Vardi. On the foundations of the universal relation model. *ACM Trans. Database Syst.*, 9(2):283–308, 1984.
- [41] Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo. A sound and complete backward chaining algorithm for existential rules. In *Web Reasoning and Rule Systems - 6th International*

- Conference, RR 2012, Vienna, Austria, September 10-12, 2012. Proceedings*, pages 122–138, 2012.
- [42] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. In *IJCAI: International Joint Conference on Artificial Intelligence*, New York, United States, July 2016.
 - [43] 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.
 - [44] 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.
 - [45] Jerzy Marcinkowski. Achilles, turtle, and undecidable boundedness problems for small datalog programs. *SIAM J. Comput.*, 29(1):231–257, September 1999.
 - [46] Stavros Cosmadakis and Paris Kanellakis. Parallel evaluation of recursive rule queries. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS ’86, pages 280–293, New York, NY, USA, 1986. ACM.
 - [47] Moshe Y. Vardi. Decidability and undecidability results for boundedness of linear recursive queries. In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS ’88, pages 341–351, New York, NY, USA, 1988. ACM.
 - [48] Irène Guessarian. Deciding boundedness for uniformly connected datalog programs. In *ICDT’90, Third International Conference on Database Theory, Paris, France, December 12-14, 1990, Proceedings*, pages 395–405, 1990.

- [49] Arash Karimi, Heng Zhang, and Jia-Huai You. Beyond skolem chase: A study of finite chase under standard chase variant. In *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017.*, 2017.
- [50] Arash Karimi, Heng Zhang, and Jia-Huai You. Restricted chase termination: A hierarchical approach and experimentation. In *Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings*, pages 98–114, 2018.
- [51] Cristina Civili and Riccardo Rosati. Bounded implication for existential rules. In *Proceedings of the 29th International Workshop on Description Logics*. CEUR Workshop Proceedings, 4 2016.
- [52] Stathis Delivorias, Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. On the k-boundedness for existential rules. In *Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings*, pages 48–64, 2018.
- [53] Stathis Delivorias, Michel Leclère, Marie-Laure Mugnier, and Federico Ulliana. On the k-boundedness for existential rules. *CoRR*, abs/1810.09304, 2018.
- [54] David Carral, Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Sebastian Rudolph. Preserving constraints with the stable chase. In *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, pages 12:1–12:19, 2018.
- [55] Swan Rocher. *Querying Existential Rule Knowledge Bases: Decidability and Complexity*. Theses, Université de Montpellier, November 2016.
- [56] 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.

- [57] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, and Andreas Pieris. A logical toolbox for ontological reasoning. *SIGMOD Rec.*, 40(3):5–14, November 2011.
- [58] 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.
- [59] George Konstantinidis. *Scalable Data Integration Under Constraints*. PhD thesis, University of Southern California, 2015.

Index

- α -operator, 22, 23
- SO-equivalence, 70
- bf-R-compliance, 119
- ancestor, 84, 90
- ancestor tree, 155
- applicability of rule
 - to factbase, 22
- applicability of trigger
 - applicability on factbase, 37
 - R-applicability on factbase, 49
 - X-applicability on derivation, 45
- boundedness, 26
 - uniform, 30
 - program, 30
 - predicate, 31
 - X-boundedness, 105
 - X- k -boundedness, 108
 - \exists -X-boundedness, 163
- chase graph, 87
 - k -minimal, 149
 - depth, 88
- chase space, 94
 - generator, 95
 - rank, 98
- chase variant, 45
 - breadth-first, 47
 - core (C), 52, 62, 64, 135
 - equivalent (E), 50, 53, 101, 150
 - frugal & vacuum (F, V), 56, 58, 77, 91, 133
 - local core (LC), 142, 149
 - oblivious (O), 50, 72, 94, 95, 97, 98, 115
 - parallel (P), 61, 82, 120
 - restricted (R), 50, 74, 75, 79, 115
 - semi-oblivious (SO), 50, 70, 72, 115, 169
- core of atomset, 19
- datalog rules, 14, 26, 29, 31, 69, 94
- depth-order independence, 137
- derivation, 38, 39
 - class of derivations, 45
 - corresponding to chase graph, 91
 - depth, 44
 - isomorphic, 147
 - monotonic, 43
 - prefix, 40
 - rank compatible, 44

- sequence of triggers (trig)*, 40
- submonotonic*, 64
- synchronous*, 60
- X-derivation*, 45
- domain of substitution, 14
- exhaustivity, 46
- factbase, 13
 - active*, 40
 - transitory*, 40
- fes (finite expansion set), 26, 34
- frontier variables (fr), 13
- fus (finite unification set), 27, 34
- generator, 154
- heredity, 115
- homomorphism, 15
- immediate derivation, 22, 37
- knowledge base, 20
- linear rules, 14, 35, 112, 162
- new variables (nulls), 23, 37
- oblivious restriction, 114
- partial core, 142, 174
- piece
 - in atomset*, 19
 - in rule head*, 166
 - in trigger output*, 56, 176
- preservation of ancestry, 110, 112, 116, 127, 133, 143
 - loose*, 149
- quasi-equivalence class, 16
- quasi-isomorphism, 16
- rank, 43, 124
- rank mark, 44, 60
- redundancy, 17, 18
 - strength in eliminating*, 67
- retraction, 18, 121
- ruleset, 13
- subsumption, 57
 - isomorphic*, 57
- termination
 - order dependence*, 66
 - all-instance (all-factbase)*, 6, 35, 112
 - of X-derivation*, 46
 - strength*, 67
- trigger, 22, 23, 37, 38, 123, 160
 - isomorphic*, 146
- universal model, 21, 22, 38, 164

Résumé en Français

On considère le formalisme des *règles existentielles* [1, 2] qui permet de représenter et de raisonner avec des connaissances ontologiques, i.e. des informations générales sur un domaine d'application. Ce sont des règles positives du type “si *corps* alors *tête*” où le corps et la tête sont des conjonctions de formules atomiques (des atomes de logique du premier ordre sans fonction). Les variables du corps sont quantifiées universellement et les variables n'apparaissant que dans la tête sont quantifiées existentiellement. Par exemple, la règle $\forall x (humain(x) \rightarrow \exists y (humain(y) \wedge mereDe(y, x)))$ représente la connaissance que chaque humain x a une mère humaine y .

Les règles existentielles étendent *Datalog* [3], le langage des bases de données déductives, et ainsi sont aussi connues comme *Datalog*⁺. En *Datalog*, toutes les variables qui apparaissent dans la tête d'une règle, apparaissent aussi dans le corps. Elles ne contiennent donc pas de variables existentielles et ne peuvent pas inférer l'existence de nouveaux individus. Un exemple simple d'une telle règle est $\forall x \forall y (frereOuSoeurDe(x, y) \rightarrow frereOuSoeurDe(y, x))$. Dans de nombreux domaines d'application, cette limitation des règles *Datalog* n'est pas satisfaisante car on ne peut pas toujours supposer que tous les individus pertinents sont connus a priori ; l'inférence de l'existence de nouveaux individus est ainsi reconnue comme une fonctionnalité désirée des langages ontologiques. Une telle fonctionnalité est offerte par de nombreux langages de représentation d'ontologies comme les logiques de description (même celles dites légères telles que *DL-Lite* [4] et \mathcal{EL} [5]) ou les règles existentielles.

Les règles existentielles sont également connues sous l'appellation *dépendance génératrice de tuples* comme des contraintes de bases de données [6], mais récemment elles ont suscité un regain d'intérêt comme langage ontologique exploité pour l'*interrogation de données médiatisée par une ontologie* (cf. par exemple les chapitres de synthèse dans [7, 8]). Dans ce contexte, une *base de connaissances* comprend un ensemble de règles existentielles (parfois appelé *ontologie*) et une *base de faits* qui est une conjonction d'atomes exis-

tentiellement fermée. La base de faits peut être vue comme une abstraction logique d’une base de données. Une *requête conjonctive Booléenne* est aussi une conjonction d’atomes existentiellement fermée. Le problème central étudié dans ce cadre est le suivant :

“étant donnée une base de connaissances K et une requête Q , K permet-elle de déduire Q ?”

Une approche classique de résolution de ce problème est connue comme la *matérialisation* ou le *chaînage avant* [9, 10]. Dans cette approche, on utilise les règles pour inférer de nouvelles connaissances en étendant la base de faits jusqu’à saturation. La requête Q peut alors être évaluée directement sur la base de faits saturée. Le problème posé par cette approche est que le chaînage avant ne termine pas sur toutes les bases de connaissances. Il est en effet connu que le problème de répondre à une requête conjonctive Booléenne est indécidable en général pour le langage des règles existentielles [11]. Ceci a conduit à identifier des sous-langages qui assurent la terminaison du chaînage avant, en imposant des restrictions syntaxiques sur la forme des règles. Par exemple, si on a la base de connaissances :

$$\begin{aligned} &\forall x \left(\text{humain}(x) \rightarrow \exists y \left(\text{humain}(y) \wedge \text{mèreDe}(y, x) \right) \right) \\ &\text{humain}(\text{Avril}) \end{aligned}$$

on peut appliquer la règle au seul atome en déduisant la base de faits étendue :

$$\exists y_0 \left(\text{humain}(y_0) \wedge \text{mèreDe}(y_0, \text{Avril}) \wedge \text{humain}(\text{Avril}) \right)$$

Ici y_0 est une variable existentielle qui est introduite pour représenter “la mère de Avril”. Maintenant la règle est applicable sur l’atome $\text{humain}(y_0)$. La nouvelle application va encore produire un nouvel individu (une variable existentielle) y_1 , relié au y_0 de la même façon que y_0 est relié à Avril, i.e. y_1 désignera “la mère de y_0 ”. De cette manière on peut créer une base de faits de taille illimitée, en représentant une chaîne d’ancêtres d’Avril. C’est évident que dans ce cas le chaînage avant ne termine pas. Toutefois il y a certains cas où l’introduction de nouvelles variables dans la base de faits n’exprime pas de nouvelle connaissance. Par exemple, si la base de faits initiale contient déjà la mère

d'Avril :

$$\textit{humain}(\textit{Pascale}) \wedge \textit{mèreDe}(\textit{Pascale}, \textit{Avril}) \wedge \textit{humain}(\textit{Avril})$$

l'application de notre règle sur l'atome $\textit{humain}(\textit{Avril})$ sera redondante. Ainsi, il est intéressant de définir des algorithmes qui vont réguler les applications des règles de telle façon que cela évite l'ajout d'atomes redondants dans la base de faits.

Les algorithmes qui effectuent le chaînage avant sont collectivement connus sous l'appellation *chase* [12, 13, 14]. Il en existe de nombreuses variantes : l'*oblivious chase* [15], le *semi-oblivious chase* (appelé aussi le *skolem chase*) [16], le *restricted chase* (appelé aussi le *standard chase*) [17], le *core chase* [18], etc. Chaque variante de chase impose des restrictions sur le critère d'application des règles et sur l'évolution de la base de faits. Le plupart des variantes de chase (dont toutes celles qui sont présentées dans cette thèse) produisent des résultats qui sont logiquement équivalents et de plus constituent un *modèle universel* de la base de connaissance, c'est-à-dire un modèle (au sens de la sémantique de la logique des prédicats) qui peut être homomorphiquement mappé sur chaque autre modèle de cette base de connaissances. Cependant l'arrêt de chaque chase diffère. Malheureusement le problème de l'arrêt du chase pour un ensemble de règles donné, que ce soit sur une base de faits donnée ou quelque soit la base de fait sconsidérée, a été démontré indécidable dans le cas général [16, 18, 1, 19]. Néanmoins, de nombreux travaux sont dédiés à la recherche de conditions suffisantes pour la *terminaison du chase* pour des sous-langages de règles existentielles [1, 20, 21, 22, 23].

Dans cette thèse on définit un cadre unificateur qui permet de présenter la plupart des variantes connues du chase. On introduit une nouvelle définition du concept de *dérivation* qui encapsule avec précision la suite des applications de règles et leur effet sur la base de faits. Cela nous permet de définir formellement la notion de *variante de chase* comme une classe spécifique de dérivation et d'introduire des propriétés facilitant la comparaison des différentes variantes ainsi que l'obtention de résultats techniques concrets.

Il est connu que les bases de faits peuvent être représentées comme des graphes/hypergraphes, et que les homomorphismes entre ces graphes correspondent à de la conséquence logique [27, 1]. Nous exploitons cette perspective “graphe” sur les règles existentielles en utilisant exclusivement la théorie des (hyper)graphes et la théorie élémentaire des ensembles pour toutes les preuves et parties techniques. En particulier, on caractérise la *redondance* dans une base de faits par l’existence d’une *rétraction* dans le graphe associé, qui est un type particulier d’homomorphisme de graphes vers l’un de ses sous-graphes. Ainsi, on met en lumière les liens entre les variantes du chase et les rétractions, puisque le principal objectif des différentes variantes est l’élimination des redondances causées par l’introduction de nouvelles variables dans la base de faits.

Le cadre unificateur de spécification de variantes du chase permet par ailleurs de définir facilement de nouvelles variantes. En étudiant les variantes connues, on montre comment de petites modifications peuvent mener à des variantes ayant des propriétés très différentes. Deux nouvelles variantes sont introduites qui optimisent, en terme d’élimination de redondances, des variantes existantes.

Une propriété en rapport avec la terminaison est celle de l’existence d’une borne à la *profondeur* du chase indépendamment de la base de faits pour un ensemble de règles donné. Les règles sont alors dites à *saturation bornée*. La profondeur d’une dérivation est le rang maximal d’un atome inféré, le rang d’un atome correspondant au nombre d’applications de règles “non-parallèles” ayant permis d’inférer cet atome. La propriété de saturation bornée entraîne plusieurs autres propriétés sémantiques, en particulier la reformulabilité en requête du premier ordre [24].

La propriété de saturation bornée a été très étudiée en Datalog [29, 30, 31, 25], et il a été prouvé que c’est une propriété indécidable dans le cas général. Par contre, pour les règles existentielles, peu de travaux ont été menés. La première chose à observer est qu’à cause des variables existentielles il faut paramétrer la propriété de saturation bornée par la variante du chase étudiée. En effet, si X et Y sont deux variantes différentes de chase, un ensemble de règles peut être X -borné (i.e. à saturation bornée par rapport au X -chase) mais pas Y -borné.

La seconde observation est liée au fait que pour chaque sous-langage de règles existentielles qui contient Datalog, on sait déjà que le problème de l'existence d'une borne à la profondeur d'un X-chase pour un ensemble de règles donné est indécidable. Dans cette thèse, on s'est donc intéressé au problème de la *X-k-saturation-bornée* où la borne k est donnée :

“étant donné un ensemble de règles et un entier k , k est-il une borne à la profondeur du X-chase quelque soit la base de faits considérée ?”

Notre approche exploite le concept de *chase graphe*. Le chase graphe est une représentation partielle d'une dérivation où les sommets sont les atomes qui apparaissent dans la dérivation et les arcs indiquent quels atomes sont utilisés pour produire d'autres atomes. L'intuition fournie par le chase graphe est utile quand on recherche les mécanismes qui influencent la saturation bornée. En particulier, on introduit une propriété de *préservation d'ascendance* qui peut être vue comme l'invariance d'une partie du chase graphe quand on réduit la base de faits initiale. On montre que si une variante X du chase préserve l'ascendance, alors le problème de la *X-k-saturation-bornée* est décidable. On montre alors que de nombreuses variantes du chase préservent l'ascendance (comme l'oblivious, le semi-oblivious et le restricted chase). Enfin, on montre que si on se restreint aux règles *linéaires*, i.e. règles dont le corps contient un seul atome, si le X-chase préserve l'ascendance, alors le problème de la X-saturation-bornée est équivalent au problème de la terminaison du X-chase indépendamment de la base de faits.

En plus des résultats précédents concernant la définition de variantes du chase et l'étude de la propriété de saturation bornée, cette thèse contient un certain nombre de résultats secondaires ainsi que de nombreuses observations et exemples qui montrent et éclairent les différents scénarios du chaînage avant sur les règles existentielles.