Guarded Existential Rules with Transitivity: A Survey

Bailey Andrew

4th Year Project Report
Artificial Intelligence and Mathematics
School of Informatics
University of Edinburgh

2021

Abstract

We survey the current state of knowledge about the compatibility of transitivity with the guarded family of rules - those deriving from the Guarded Fragment of First Order Logic. Compatibility is defined using three different notions; the standard notion and two restricted notions frequently considered in the literature (transitive guards and base guards). We then examine each notion of compatibility for the three standard query types (Atomic, CQ, UCQ), contributing several original results in the process. These original results modestly improve our understanding of the important case of compatibility of Linear rules with transitivity, as well as complete the picture for Guarded Disjunctive and Frontier Guarded rules and some of their superclasses.

Acknowledgements

I would like to thank my supervisor, Andreas Pieris, for their support in the creation of this report.

Table of Contents

Chapter 1

Introduction

Databases are an integral part of the modern digital ecosystem, and much effort has been invested in their study. In this report we are interested in the Ontology-Based Data Access (OBDA) paradigm, in which we equip databases with a set of deductive rules (an 'ontology') so that it can derive new facts from those present in the database. Of course, if we allowed arbitrary ontologies, one could phrase undecidable problems in the language of the database; an important task in this field is designing classes of ontologies which maximize expressive power while maintaining decidability.

One notable decidable class of rules is Datalog, which while efficient suffers from the limitation that it can only reason about entities already in the database. One way around this is to add existential quantification to the language of Datalog; these rules are called tuple-generating dependencies[1] (TGDs). With existential quantification, TGDs can reason about objects they know exist without knowing which, if any, object in the database they correspond to. The extension to TGDs comes at the price of undecidability. To mitigate this, research has been done in defining decidable subclasses of TGDs, such as Linear and Guarded rules (defined in Section 2.2).

Another approach to OBDA is to consider decidable fragments of first-order logic (FOL). One such fragment is the Guarded Fragment (GF), a fragment which is capable of expressing modal logic[ANB98]. There are many subclasses of TGDs, such as the aforementioned Guarded rules, which can be interpreted as both a set of TGDs and a fragment of GF. While expressive, these rules lack the capability to express transitivity (rules of the form $T(x,y) \wedge T(y,z) \longrightarrow T(x,z)$), an important property needed to adequately capture the notions of `ancestor-of`, `less-than`, and `part-of`, among other relations. This limitation hampers their applicability in domains such as medicine, where a gene might be `part-of` the pathway for a protein that is `part-of` the regulation of a hormone that is `part-of` the circulatory system (for example).

To combat this limitation, work has been done to determine whether and to what extent transitivity can be integrated with these ruleclasses while preserving decidability, such as in [Bag+15], [GPT13], and [Ama+18]. In this report, we aim to analyze and compile these results in a central place, as well as contribute our own results. For cases

[1] These are also known as existential rules, Datalog$^{\pm}$, and $\forall\exists$-rules.

where compatibility is not yet known, we will also attempt to give arguments as to why standard approaches to the problem will not be sufficient.

1.1 Motivation

Many useful real-world constructs can be described using the transitive property, as described in the previous section. Thus, it is no wonder that we wish to understand its compatibility with decidable ontologies. In this report, we limit ourselves to results concerning compatibility with the Guarded Fragment and its relatives. This decision was made partially to make this report's scope manageable, but it was also made due to practical concerns. Most classes of ontologies are defined in one of two ways; via a syntactic condition, or via some kind of acyclicity notion. GF and most of its relatives are defined syntactically, whereas a ruleclass such as aGRD[2] is defined using acyclicity. Baget et al. have shown that many acyclically defined classes are incompatible with transitivity[Bag+15], which should not come as a surprise as there is no reason to expect that transitivity would respect the graph structure over which the ruleclasses are defined. It is hoped that syntactically-defined ruleclasses will have better luck on this front.

When considering whether or not transitivity is compatible with an ontology, there are multiple notions of 'compatibility' that one could consider. One could consider whether or not transitivity is compatible without any restrictions, but this will typically lead to undecidability. To mitigate this, we could impose restrictions on the non-transitive portions of the ontology, such as limiting ourselves to two-variable or monadic fragments (as done in [GPT13]). These types of restrictions have achieved some positive decidability results. Another method is to impose restrictions on when transitive predicates can be used; this method has achieved some success in the past, and a recent result by Amarilli et al.[Ama+18] used one such restriction to achieve a positive decidability result for a broad swathe of ontologies. Due to this recent success, we will focus on restrictions of the latter form in this report. Specifically, for a class of rules $\mathcal{X}$, we focus on the problem of general compatibility "$\mathcal{X} + \mathit{trans}$", as well as the restricted forms Base$\mathcal{X} + \mathit{trans}$ and $\mathcal{X} + \mathrm{TG}$; their definitions are deferred to Chapter 2.

The final decision that was made in determining the scope of this project was that of which query type to consider. We limit ourselves to atomic, conjunctive, and unions of conjunctive queries. These three query types are standard in the field; there are others that are used, such as acyclic queries, however the scope of this project is already sufficiently broad that we saw no need in including additional query variants.

[2] aGRD stands for 'acyclic graph of rule dependencies', a class which seeks to disallow rules from triggering themselves. As the name implies, it is defined using an acyclicity notion, where the relevant graph is that of rules connected by edges if one rule can trigger another.

1.2 Overview of Contributions

While this report is primarily a survey, it contains some original results as well. Original results will be explicitly labeled as either a **Theorem** or a **Corollary** when they are introduced, or will otherwise be explicitly noted as original. Additionally, all results in the appendix are original. In this section, we will give a brief list of results original to this report.

1. **Theorem 1**: Linear + TG is decidable for CQs.
2. **Corollary 1**: GDTGD + TG is undecidable for CQs.
3. **Corollary 2**: Linear + TG is decidable for CQs, even if we allow the head to be non-atomic.
4. **Theorem 2**: GNF + TG is undecidable for atomic queries (in fact even GNF + TG satisfiability is undecidable).
5. **Theorem 3**: FGTGDs + TG is undecidable for atomic queries.

Additionally, there are some original results about binary cross products (rules of the form $C_1(x) \wedge C_2(y) \longrightarrow \bar{C}(x,y)$) in the appendix. Like transitivity, cross products are a useful construct that cannot be expressed within the syntactic conditions of GF. We found it useful to reference cross product results for comparisons with results for transitivity, although they have been relegated to the appendix as they are not relevant enough to deserve integration into the main body of the report.

In the opinion of the author, the original results achieved for this survey are mostly incremental improvements on the current state of knowledge in the field. While nontrivial, they do not represent the kind of fundamentally new step forward that would be required to resolve some of the remaining open problems, such as the CQ-decidability of Linear+*trans*. Despite this, the results here were enough to complete the decidability picture of GDTGDs, FGTGDs, GF, and GNF, tightening the decidability frontier significantly; before this survey, a complete picture for all three compatibility types and all three query types had not been attained for any of the rulesets considered in the report.

Chapter 2

Background

Before presenting the survey of results, we will give formal definitions to the notions discussed in the introduction. We have split this section into two parts; the first of which will cover the general problem we are attempting to solve, and the second will cover the types ruleclasses that we will be considering. We assume familiarity with logical notation, and will frequently use the concepts 'term' (a variable or a constant), and 'atom' (a single non-negated predicate with some terms, i.e. $p(x,y)$).

2.1 The Query Answering Problem

Databases in the OBDA paradigm can be thought of as a tuple containing a set of initial facts $\mathcal{F}_0$ (corresponding to the traditional notion of a database as merely a collection of data), and a set of rules $\mathcal{R}$. We are concerned with the Query Answering problem; given a set of facts $\mathcal{F}_0$, a set of rules $\mathcal{R}$, and a query $Q(\vec{x})$ with some free variables $\vec{x}$, what is the largest set of tuples such that for each element in the set $\vec{c}$ we have that $\mathcal{F}_0, \mathcal{R} \models Q(\vec{c})$? As an example, here is an instance of the problem;

Facts $\mathcal{F}_0$

$\texttt{mother\_of}(\texttt{Alice}, \texttt{Bob})$

$\texttt{mother\_of}(\texttt{Clarissa}, \texttt{Alice})$

Rules $\mathcal{R}$

$\texttt{mother\_of}(x,y) \wedge \texttt{mother\_of}(y,z) \longrightarrow \texttt{grandmother\_of}(x,z)$

Query $Q(x)$

$\texttt{grandmother\_of}(x, \texttt{Bob})$

Problem:

What are all the x such that $\mathcal{F}_0, \mathcal{R} \models Q(x)$?

Every instance of this problem can be rewritten as a yes-or-no entailment problem[Bag+15] without free variables: $\mathcal{F}_0, \mathcal{R} \models Q$. This is known as the 'Boolean' variant of the problem. As the two variants of the problem are equivalent, we only consider the Boolean

variant of the problem in this report. Query Answering can also be written as a satisfiability problem: $\mathcal{F}_0, \mathcal{R} \models Q \iff \mathcal{F}_0, \mathcal{R}, \neg Q \models \bot$. This typically does not preserve ruleclasses - for a class of rules $\mathcal{C}$, $\mathcal{R} \in \mathcal{C} \not\longrightarrow \mathcal{R} \wedge \neg Q \in \mathcal{C}$. Despite this, the reduction to satisfiability can be helpful, so it will be useful to keep this in mind.

Formally, the factbase $\mathcal{F}_0$ is rather simple; it is a conjunction of atoms containing only constants. The form of the ruleset $\mathcal{R}$ in contrast can be very complex, and will depend on the ruleclass chosen - specific examples will be defined in Section 2.2. Like $\mathcal{R}$, we have a few choices for the form of the query Q; however, it should be clear that we need to be careful if we wish to preserve decidability; even if we had an empty ruleset and an empty factbase, a sufficiently complicated query would be undecidable. This is not hard to see, as the corresponding satisfiability problem, $\neg Q \models \bot$, could be made arbitrarily complex with a sufficient choice of Q.

In this report we consider three types of queries; atomic queries, conjunctive queries (CQs), and unions of conjunctive queries (UCQs). Atomic queries are queries of the form $\exists \vec{x}.q(\vec{x}, \vec{C})$, where $\vec{C}$ are constants. CQs are queries of the form $\exists \vec{x} \bigwedge_i q_i(\vec{x}_i, \vec{C}_i)$, and UCQs have the form $\exists \vec{x} \bigvee_i \bigwedge_j q_{ij}(\vec{x}_{ij}, \vec{C}_{ij})$. Intuitively, CQs are conjunctions of atomic queries, and UCQs are disjunctions of CQs. As a shorthand, if UCQ query answering for a certain class of rules $\mathcal{C}$ is undecidable, then we will write that $\mathcal{C}$ is UCQ-undecidable (and likewise for atomic queries and CQs).

As the Query Answering problem takes three inputs $(\mathcal{F}_0, \mathcal{R}, Q)$, there are a few ways to measure computational complexity. The most immediate is 'combined complexity', in which we measure the complexity in terms of the size of all three inputs taken together. However, in real world applications the ruleset is likely to be fixed and the queries are likely to be small - most change to the ontology will come from adding new facts to the dataset $\mathcal{F}_0$. 'Data complexity' attempts to address this; it only measures the complexity as a function of the size of $\mathcal{F}_0$, considering the other two inputs to be fixed. There are other complexity measures, such as bounded-arity combined complexity (which also takes into consideration the arity of predicates), but in this report we only consider (unbounded-arity) combined complexity and data complexity.

There is a nuance in how we measure complexity for the Query Answering problem, which is important helpful to mention. We can measure the data complexity for a ruleset; "how does the size of the factbase affect the runtime for the ruleset $p(x) \longrightarrow q(x)$ and query $q(\texttt{Bob})$?" is a well-founded question. However, we are not interested in the complexity of rulesets; rather, we are interested in the complexity of classes of rules. It might be possible a Guarded Fragment ruleset has AC_0 data complexity, but that does not mean that GF is AC_0 (in fact it is coNP-complete). In essence, when talking about the complexity of a ruleclass, we are actually asking about the maximum complexity out of all possible instances of the problem which are in that ruleclass.

2.2 The Guarded Fragment and Its Subclasses

The general form of a TGD is $\forall\vec{x},\vec{y}.\bigwedge_i B_i(\vec{x},\vec{y}) \longrightarrow \exists\vec{z}.\bigwedge_j H_i(\vec{x},\vec{z})$, where B_i, H_i are atoms, and $\vec{x},\vec{y},\vec{z}$ are sets of terms. Universal quantification is typically omitted, as it can be assumed that all variables not existentially quantified are universally quantified. The portion of the sentence to the left of the implication is known as the 'body', and the portion to the right is known as the 'head'. Variables appearing in both the body and the head (those in $\vec{x}$) are known as 'frontier variables'.

In this report, we primarily consider Linear and Guarded TGDs (GTGDs). Both are defined in terms of a syntactic condition. A TGD is Linear if both its body and head consist of a single atom, and a rule is Guarded if there is an atom in the body which contains all terms present in the body - such an atom is known as the 'guard'. $p(x,y) \longrightarrow h(x,z)$ is Linear and Guarded (in fact, all Linear rules are trivially Guarded), while $p(x,y) \wedge q(x) \longrightarrow h(x,y)$ is Guarded but not Linear. $p(x,y) \wedge q(z) \longrightarrow h(x,y)$ is neither Guarded nor Linear, as there is no body atom which contains all variables x,y,z.

In this report, we will also investigate certain types of disjunctive TGDs (DTGDs), in which we allow disjunction in the head. Disjunctive Linear and Disjunctive Guarded rules (D-Linear and GDTGDs) are defined analogously to their non-disjunctive variants; a ruleset is D-Linear if the body is atomic and the head is a disjunction of atoms, and a ruleset is a GDTGD if the body is guarded and the head is a disjunction of conjunctions of atoms. $p(x,y) \longrightarrow \exists z.h(x,z) \vee p(x,z)$ is D-Linear, but $p(x,y) \longrightarrow \exists z.h(x,z) \vee (p(x,z) \wedge h(z,z))$ is not, as there is a conjunction in the head. All Linear rules are D-Linear, and all Guarded rules are GDTGDs.

Both Linear and Guarded rules, and their disjunctive variants, are subclasses of the Guarded Fragment[1]. GF ontologies are not TGDs; rather, we say that a sentence in FOL is GF if it satisfies a certain set of syntactic restrictions. These restrictions are most easily defined inductively, as the smallest set such that:

1. All atoms are in GF.
2. All equalities are in GF (i.e. $x = y$ is in GF)
3. If ϕ,ψ are GF, then $\phi \wedge \psi, \phi \vee \psi, \phi \longrightarrow \psi$, and $\neg\psi$ are all GF.
4. $\forall\vec{x}.\psi \longrightarrow \phi$ and $\exists\vec{x}.\psi \wedge \phi$ are GF if ψ is an atom containing all variables in $\vec{x}$ as well as all free variables in ϕ. In this case, ψ is the 'guard'.

As an example, $\neg\forall x.G(x) \longrightarrow P(x) \wedge Q(x)$ is guarded, with G being a guard. However, $\neg\forall x.G(x) \longrightarrow \exists yzv.P(z,y) \wedge Q(v,z)$ is not, as there is no guard for the existential.

While GF allows equalities, in this report we will consider only equality-free GF sentences. We decided to remove equalities because the majority of results in this survey are undecidability results. Including equalities would increase the expressive power of a ruleclass, and thus threaten to further increase the amount of undecidability results. Additionally, it is not immediately clear if the work done by Amarilli et al,

[1] Technically, Guarded rules do not require a guard for existential quantifiers, while GF does - however it is trivial to introduce a 'dummy atom' in the head of a rule to serve as the guard, in which case the rule becomes GF.

who have produced a number of powerful decidability results in an equality-free context[Ama+18], would extend when equality is considered.

One other type of TGD are the 'inclusion dependencies' (IDs), and their disjunctive variant (DIDs). They are a subclass of Linear rules in which no variable occurs twice in the same atom. We will not explicitly consider this class in this survey, and instead note that all (un)decidability results for Linear rules given in this survey also apply to IDs (and likewise for the disjunctive variants)[2]. Complexity results may differ, however.

This report is focused on the compatibility of transitivity with (D-)Linear, (D-)Guarded, and GF rules. There are three ways in which we check whether or not transitivity is compatible with a ruleset, which can be summed up with the labels 'general transitivity', 'transitivity only in guards', and 'transitivity only outside of guards'. We will explain these rules in terms of the UCQ decidability of GF first, and use the notation $T_i^{\mathcal{R}}$ to denote the rule expressing the transitivity of T_i ($T_i(x,y) \wedge T_i(y,z) \longrightarrow T_i(x,z)$) for brevity.

1. General transitivity: We say that "GF+*trans*" is UCQ-decidable if the Query Answering problem can be decided for UCQs for rulesets of the form $\phi \wedge \bigwedge_i T_i^{\mathcal{R}}$, where ϕ is a GF ruleset and there are no restrictions on where the predicates T_i can appear in ϕ
2. Transitivity only in guards: We say that "GF+TG" is UCQ-decidable if the Query Answering problem can be decided for UCQs for rulesets of the form $\phi \wedge \bigwedge_i T_i^{\mathcal{R}}$, where ϕ is a GF ruleset and T_i only appear as guard atoms in ϕ. TG stands for *Transitive Guards*.
3. Transitivity only outside of guards: We say that "BaseGF+*trans*" is decidable using the same setup as in the previous two definitions, but instead requiring T_i to never appear as guard atoms in ϕ. (Atoms which are not transitive are known as 'base atoms', which is why this form is called *Base*GF+*trans*; only base atoms are allowed to appear in guards).

Defining CQ- and atomic-decidability is completely analogous. Extending this definition to work for GTGDs and Linear rules is also analogous, although we have to be careful with how we define 'guards'; even though GTGDs have their own (related) concept of what a guard is, GTGD+TG restricts transitive predicates to appearing only in atoms which are guards *in the sense of the Guarded Fragment*. There are a few subtleties in this definition, so it will help to see some examples. In the following examples, T always represents a transitive predicate.

1. $T(x,y) \longrightarrow H(x,y)$ is Linear+TG, as T appears in the guard. However, it is not BaseLinear+*trans*, for the same reason.
2. $T(x,y) \wedge P(x,y) \longrightarrow H(x,y)$ is both Linear+TG and BaseLinear+*trans*. This is because either T or P could serve as the guard for this rule; we get to choose,

[2]To see why this is, it is sufficient to note that all decidability results for Linear rules trivially prove decidability for IDs, and that all undecidability results given in this report will use a ruleset that is either an ID, or is rewritable as an ID without much work.

so if we want the ruleset to be Linear+TG, we can let T be the guard, and if we want it to be BaseLinear+*trans*, we can let P be the guard.

3. $T(x,y) \wedge T(y,x) \longrightarrow H(x,y)$ is neither Linear+TG nor BaseLinear+*trans*, as no matter which T we choose as the guard, the other T is not the guard.

4. $B(x,y) \longrightarrow T(x,y)$ is BaseLinear+*trans*, but not Linear+TG, as T does not appear in a guard.

5. $B(x,y) \longrightarrow \exists z.T(x,z)$ is Linear+TG, as T guards the existential quantifier. At face value, this ruleset is not BaseLinear+*trans*, but if we introduce the dummy predicate `Aux` we can phrase it as the equivalent ruleset $B(x,y) \longrightarrow \exists z.T(x,z) \wedge \texttt{Aux}(z)$. Now, we can let `Aux` be the guard, and hence it is BaseLinear+*trans*. The introduction of an auxiliary predicate is trivial, and thus we will always handle this implicitly, allowing us to say that $B(x,y) \longrightarrow \exists z.T(x,z)$ is BaseLinear+*trans*.

The justification behind considering TG and Base- variants of the transitivity compatibility problem is that transitivity tends to lead to undecidability, whereas these variants are able to limit transitivity enough into something more manageable in a manner which arises naturally from the definition of the Guarded Fragment.

Chapter 3

Decidability Results

In this chapter, we will perform a case-by-case survey on the ruleclasses introduced in Section 2.2. Each section in this chapter will pertain to one of these ruleclasses, and will be split into up to three sub-sections, one for each of the three query types (atomic/CQ/UCQ). Some classes will have less sub-sections, as the results for one type of query may follow trivially from the results for another. Each section will contain a table outlining the three types of compatibility results (general, TG, Base-) for each of the three queries, as well as a discussion as to how those results were obtained, along with proofs if the results are original. This chapter is mostly ordered by increasing the expressive power of the ruleclasses. However, Linear rules (the least expressive) are left until the end, as the results for Linear rules are the most technically involved. Figure 3.1 contains a graphical overview of the current decidability frontier for CQs.

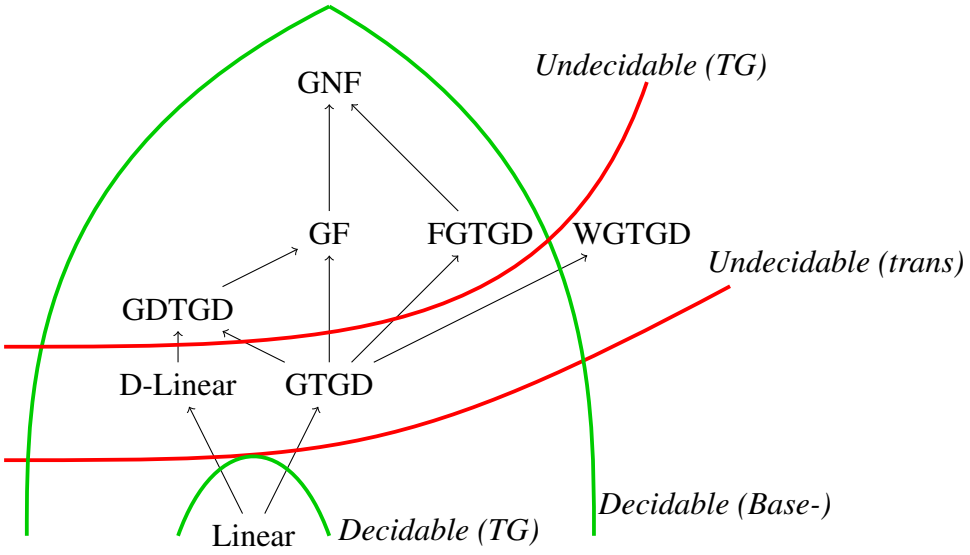

Figure 3.1: The current decidability frontier for CQs. Classes above red lines are undecidable, and classes below green lines are decidable (for the relevant compatibility type). Note that there are no known negative decidabilty results for Base-, and no known positive decidability results for general transitivity (for CQs). Arrows represent inclusion; the arrow from Linear to GTGD represents the fact that Linear rules are GTGDs.

3.1 Disjunctive Linear Rules

Disjunctive Linear Rules	+Trans	+TG	Base+Trans
Atomic Queries	?	Yes[ST01]	Yes[Ama+18]
CQs	No[Ama+18]	?	Yes[Ama+18]
UCQs	No[Ama+18]	?	Yes[Ama+18]

Table 3.1: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

3.1.1 Conjunctive Queries

Amarilli et al. have shown that D-Linear+*trans* is undecidable, by a reduction to a tiling problem[Ama+18]. Specifically, they reduce to the following problem; given a finite set of colored tiles, and finite sets of vertical/horizontal 'forbidden pairs', can the first quadrant of the plane be tiled without having any two colors in a horizontal (resp. vertical) forbidden pair appearing horizontally (resp. vertically) adjacent to each other?[1] Order matters; red being above green may be forbidden while green being above red may be allowed.

The proof we use in this paper is a modified version of theirs; the modifications were made with the aim of making it easier to explain. Our version also lends itself well to a nice result about cross products, in Appendix A.2.

Suppose we had the following D-Linear+*trans* ruleset:

$$
\begin{aligned}
S'(x,y) &\longrightarrow \exists z.S'(y,z_{\exists}) \\
S'(x,y) &\longrightarrow S^{+}(x,y) \\
S^{+}(x,y) &\longrightarrow \bigvee_{i} K_i(x,y) \\
S^{+}(x,y) \wedge S^{+}(y,z) &\longrightarrow S^{+}(x,z) \text{ (transitivity)} \\
S^{+}(x,y) &\longrightarrow S^{+}(y,x) \text{ (symmetry)} \\
\bigwedge_{0\le i<n} S'(a_i,a_{i+1}) \wedge \bigwedge_{0\le i\le n} K_j(a_0,a_i) &\text{ (facts)}
\end{aligned}
$$

In this ruleset, $K_j(x,y)$ will represent that the tile at grid coordinates (x,y) has color C_j. Our terms x intuitively represent natural numbers, with $S'(x,y)$ stating that $y = x+1$. Our first rule creates an infinite chain of S', representing the number line. Without the symmetry rule, S^{+} would represent the relation $<$. The addition of the symmetry rule allows it to represent the Cartesian product of the natural numbers with themselves, i.e. it represents the coordinate pairs of the first quadrant of the plane. Finally, the disjunctive rule forces each tile to choose a color.

[1] Additionally, there is the requirement that tiles at positions $(i,0)$ where $i <$ # of colors must be colored as color C_{i-1}

It is not hard to see that, for a horizontal forbidden pair (C_i, C_j), the query $S'(x,x') \wedge K_i(x,y) \wedge K_j(x',y)$ checks if it is violated. We can make similar conjunctions for every forbidden pair, and consider the UCQ formed by their union; the UCQ will be entailed if and only if at least one forbidden pair is violated, which completes the reduction.

Amarilli et al. go on to present a method to transform the ruleset in a manner that shows CQ entailment is undecidable as well, by increasing the arity of certain the predicates. The general idea of the method is well-known in the field, although they made the realization that not all predicates need to increase in arity - importantly, this means that the transitive predicate does not need to change.

They do this by adding two constants $\mathfrak{t}, \mathfrak{f}$ (intuitively representing 'true' and 'false'), along with a ternary predicate `Or`, which encodes the truth table of disjunction (i.e. the set of initial facts contains atoms of the form $\texttt{Or}(x,y,x \vee y)$ where $x,y \in \{\mathfrak{t},\mathfrak{f}\}$). They are then able to encode the disjunctive essence of the query into a conjunction involving the `Or` predicates.

Traditionally, this method would require increasing every predicate's arity, but in fact they only need to increase the arity of S', as it appears in every disjunct and is the 'entry point'[2] into the ruleset.

This result is impressive, as the ruleset itself is incredibly simple; it contains one existential rule, one disjunctive rule, one transitivity rule, and two datalog rules, all of which are linear rules (and for the UCQ case are all of arity 2 as well). The simplicity of the ruleset implies that it will be hard to find any further restriction that would allow transitivity to become compatible with any type of disjunctive rule[3].

[2] 'Entry point' means that the only type of atom that can derive an S' atom is another S' atom. If another atom were to derive S', we would also need to increase its arity to be able to propagate the term in the third position, which is always inherited from the initial set of facts and is either $\mathfrak{t}$ or $\mathfrak{f}$.

[3] Hard, but not impossible, as in Section 4.1 we will see that GNF, a superclass of D-Linear, is compatible with transitive rules with the restriction that transitive predicates never appear in guards.

3.2 Guarded Rules

Guarded Rules	+Trans	+TG	Base+Trans
Atomic Queries	No[GPT13]	Yes[ST01]	Yes[Ama+18]
CQs	No[GPT13]	?	Yes[Ama+18]
UCQs	No[GPT13]	?	Yes[Ama+18]

Table 3.2: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

3.2.1 Atomic Queries

Atomic query answering with general transitivity was shown to be undecidable by Gottlob et al.[GPT13]. To do this, they constructed a grid using transitive relations, and then encoded a Turing machine over that grid; the query then asks whether that machine ever enters the halting state. We can get an intuitive feel for how it works by watching a timelapse of how it might generate a grid, in Figure 3.2.

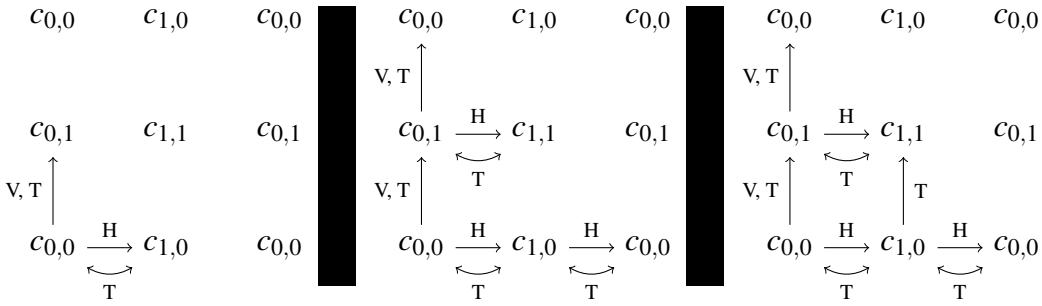

Figure 3.2: From left to right, the initial way in which a grid is built up. Grid coordinates are represented by a single term x (compared to the D-Linear proof where they were represented by a pair (x,y)); all grid terms make one of the four $c_{i,j}$ predicates true, which are used to allow the ontology to understand the local landscape of the grid.

It is unlikely that this method will be extendable to Guarded+TG. In such a case, only a single transitive relation may be created at a time (otherwise there would have to be more than one guard in the head, a contradiction), and they must be created at the same time as the grid relations V and H (if it were created after, it would not contain an existential variable). However, this grid creation method sometimes creates two transitivity atoms at the same time, such as when H is created, and it seems that alternative grid strategies share this limitation.

In fact, we can give a brief argument as to why this limitation must persist. The most straightforward trick used to build a grid using transitivity is to build the transitive relations into 3 of the edges of a grid square, and use the transitive property to derive the missing edge. This is easily doable, but suffers from the problem that transitivity is very ‘reactive’ - not only will the missing edge be derived, but many other connections will be made by the transitive rule. This can be a major problem, as we would need some way of identifying which transitive predicates are ‘real’ and which are just

byproducts of the reactivity. We can try to circumvent this by using multiple transitive predicates T_i, making sure that adjacent grid squares use different transitive predicates to limit the creation of unwanted edges. However, since we need to create T_i on three edges of each square, we are limited in the ways we can build the grid. In fact, it is easy to convince oneself that there is no such way to build the grid.

$$\begin{array}{ccccccc}
(0,2) & \xrightarrow{T_2} & (1,2) & \xrightarrow{T_1} & (2,2) & & (3,2) \\
T_2\uparrow & & \downarrow T_2 & & \downarrow T_1 & & \\
(0,1) & \xleftarrow[T_1]{} & (1,1) & \xleftarrow[T_1]{} & (2,1) & \xrightarrow{T_1} & (3,1) \\
\downarrow T_1 & & \downarrow T_2 & & T_2\uparrow & & \downarrow T_1 \\
(0,0) & \xrightarrow{T_1} & (1,0) & \xrightarrow{T_2} & (2,0) & \xleftarrow[T_1]{} & (3,0)
\end{array}$$

Figure 3.3: A demonstration of the inevitable failure of attempts to build a grid with the following three criteria: (i) at most one T_i is directly created per edge, (ii) tiles have exactly three created edges for the relevant T_i, (iii) adjacent tiles use differing T_is. It is impossible to extend the given tiling to the square with (3,2) in its upper right corner, since it needs three T_2 edges but two of the edges are taken by T_1.

Additionally, it is known that atomic query answering for GF+TG, and hence GTGD+TG, is decidable, as there is a straightforward reduction to the known-to-be-decidable satisfiability problem for GF+TG[ST01]; thus, any grid method meant to show undecidability would have to be done in a way that forces the query to be at least conjunctive. This is not necessarily an insurmountable obstacle, as a grid method was done to show that GF+TG is UCQ-undecidable[GPT13].

3.3 Disjunctive Guarded Rules

Disjunctive Guarded Rules	+Trans	+TG	Base+Trans
Atomic Queries	No[GPT13]	Yes[ST01]	Yes[Ama+18]
CQs	No[GPT13][Ama+18]	No	Yes[Ama+18]
UCQs	No[GPT13][Ama+18]	No	Yes[Ama+18]

Table 3.3: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

3.3.1 Conjunctive Queries

Most results from Disjunctive Guarded Rules follow naturally from other results in this survey, however the CQ and UCQ-decidabilities for D-Linear+TG do not currently follow from the literature. In this section, we will use another modification of Amarilli et al.'s proof of D-Linear CQ undecidability to prove that GDTGD+TG must be undecidable. As this result depends heavily on that of Amarilli et al, it has been labeled as a corrollary to their result.

Corollary 1. *GDTGD+TG is CQ-Undecidable.*

Proof. Recall that the modified version of their proof given in Section 3.1 was as follows:

$$
\begin{aligned}
S'(x,y) &\longrightarrow \exists z.S'(y,z_{\exists}) \\
S'(x,y) &\longrightarrow S^{+}(x,y) \\
S^{+}(x,y) &\longrightarrow \bigvee_{i} K_i(x,y) \\
S^{+}(x,y) \wedge S^{+}(y,z) &\longrightarrow S^{+}(x,z) \text{ (transitivity)} \\
S^{+}(x,y) &\longrightarrow S^{+}(y,x) \text{ (symmetry)} \\
\bigwedge_{0\leq i<n} S'(a_i,a_{i+1}) \wedge \bigwedge_{0\leq i\leq n} K_j(a_0,a_i) &\text{ (facts)}
\end{aligned}
$$

The only situation where S^{+} is not a guard is in the second rule, and the symmetry rule. It is not hard to remove the symmetry from S^{+}; we can compensate by expanding the disjunction that introduces K_i into three rules like so:

$$
\begin{aligned}
S'(x,y) &\longrightarrow \exists z.S'(y,z_\exists)\\
S'(x,y) &\longrightarrow S^+(x,y)\\
S^+(x,y) &\longrightarrow \bigvee_i K_i(x,y)\\
S^+(x,y) &\longrightarrow \bigvee_i K_i(y,x)\\
S^+(x,y) &\longrightarrow \bigvee_i K_i(x,x)\\
S^+(x,y) \wedge S^+(y,z) &\longrightarrow S^+(x,z) \text{ (transitivity)}\\
\bigwedge_{0\leq i<n} S'(a_i,a_{i+1}) \wedge &\bigwedge_{0\leq i\leq n} K_j(a_0,a_i) \text{ (facts)}
\end{aligned}
$$

We then note that this ruleset is equivalent to the following, where we have modified the first rule and removed the second:

$$
\begin{aligned}
S'(x,y) &\longrightarrow \exists z.S'(y,z_\exists) \wedge S^+(y,z_\exists)\\
S^+(x,y) &\longrightarrow \bigvee_i K_i(x,y)\\
S^+(x,y) &\longrightarrow \bigvee_i K_i(y,x)\\
S^+(x,y) &\longrightarrow \bigvee_i K_i(x,x)\\
S^+(x,y) \wedge S^+(y,z) &\longrightarrow S^+(x,z) \text{ (transitivity)}\\
\bigwedge_{0\leq i<n} (S'(a_i,a_{i+1}) \wedge S^+(a_i,a_{i+1})) \wedge &\bigwedge_{0\leq i\leq n} K_j(a_0,a_i) \text{ (facts)}
\end{aligned}
$$

It should be clear that S^+ only appears in guards[4]. We can see that it is equivalent to the first ruleset - the main difference is that S^+ is created immediately along with S'. Without the now-removed second rule we would not generate the S^+ facts that do not contain existential variables - to fix this, we have added those facts to the initial set of facts.

As mentioned in the section 3.1, the key properties that allowed us to go from UCQ answering to CQ answering were that S' appears in every disjunct, and that S' is the entry point to our ruleset. Neither of these facts have changed; it is not hard to show that we can modify the ruleset to use a CQ, at the cost of increasing the arity of S'. Thus GDTGD+TG is undecidable for conjunctive query answering. □

[4]In fact, this ruleset is nearly D-Linear+TG; but the atomic-headedness of D-Linear prohibits this.

3.4 Guarded Fragment

GF	+Trans	+TG	Base+Trans
Atomic Queries	No[Grä98]	Yes[ST01]	Yes[Ama+18]
CQs	No[Grä98]	No	Yes[Ama+18]
UCQs	No[Grä98]	No[GPT13]	Yes[Ama+18]

Table 3.4: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

3.4.1 Atomic Queries

There is a straightforward, well-known reduction from atomic query answering with GF+TG to GF+TG satisfiability, which is known to be decidable[ST01]. Here we will briefly demonstrate this reduction.

Remark 1. *GF+TG atomic-decidability can be reduced to GF+TG satisfiability.*

Proof. If Q is a single atom $q(\vec{x},\vec{C})$ (which may or may not be transitive), then we can add a (guaranteed-to-be-guarded) rule $\mathcal{R}' = q(\vec{x},\vec{y}) \longrightarrow \texttt{Aux}(\vec{x},\vec{y})$, and instead query for $\texttt{Aux}(\vec{x},\vec{y})$. As mentioned in Chapter 2, $\mathcal{F}_0, \mathcal{R}, \mathcal{R}' \models \texttt{Aux} \iff \mathcal{F}_0, \mathcal{R} \wedge \mathcal{R}' \wedge \neg\texttt{Aux} \models \bot$. This completes the reduction, as $\mathcal{R} \wedge \mathcal{R}' \wedge \neg\texttt{Aux}$ is GF+TG. □

3.4.2 Unions of Conjunctive Queries

Using a similar grid method to the one discussed in Section 3.2, Gottlob et al. were able to show that the guarded fragment with transitive guards was undecidable for UCQ answering[GPT13]. The intuition behind the construction of the grid is very similar. A notable difference is that their construction uses only a single transitive relation; by utilizing the additional expressive power of GF, they are able to circumvent the 'reactivity' problem mentioned in Section 3.2, by detecting which grid connections are correct and which are extraneous.

3.5 Linear Rules

Linear Rules	+Trans	+TG	Base+Trans
Atomic Queries	Yes[Bag+15]	Yes[Bag+15][ST01]	Yes[Bag+15][Ama+18]
CQs	?	Yes	Yes[Ama+18]
UCQs	?	?	Yes[Ama+18]

Table 3.5: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

3.5.1 Atomic Queries

The decidability of atomic query entailment with Linear+*trans* ontologies was proven by Baget et al.[Bag+15]. To do this, they construct an algorithm that works roughly as follows:

1. For every transitive predicate T, create a 'pattern' P_T. Intuitively, the patterns will represent T and all atoms that can derive T using a sequence of non-transitive rules. For example, the following ruleset:

$$\begin{aligned} T(x,y) &\longrightarrow P(x,y) \\ Q(x,y,z) &\longrightarrow T(y,x) \\ R(x,y) &\longrightarrow \exists z.T(x,z_{\exists}) \end{aligned}$$

   would have the pattern $P_T[\#1,\#2] := T(\#1,\#2)|Q(\#2,\#1,z_0)$, since Q can derive T. Note that R is not added to the pattern; this is because there is the additional restriction that both variables in T must be present in R - but T contains an existential, so it is not included.

2. In the body of every rule, replace instances of T with the 'repeatable pattern' P_T^+; also replace instances of T in the query. Like patterns, repeatable patterns represent the atoms that can derive T - they are an extension of that idea, meant to capture the ways in which the transitivity rule itself can create new T atoms. A repeatable pattern $P_T^+[x,z]$ would be equivalent to some finite chain $P_T[x,y_0] \wedge P_T[y_0,y_1] \wedge ... \wedge P_T[y_k,z]$. A key insight in the paper is that there is an upper bound on the lengths of chains that need to be considered.

3. They then perform backwards chaining on the query until it reaches a fixed point - in doing so, they are forced to consider a few cases on how to rewrite the query 'within' a repeatable pattern, which may potentially increase the size of the query.

4. Finally, they note that the pattern definitions can be turned into a Datalog ruleset Π, and that the set of rewritings of the query can be considered as a UCQ Q, for which the problem of whether $\Pi \models Q$ is equivalent to the original problem.

The algorithm is sound and complete, but it is not guaranteed to terminate in general - it may be that the backwards chaining on the query never reaches a fixed point. However,

they show that for atomic queries, the rewriting step will never increase the size of the query, and hence it must terminate.

By focusing on backwards chaining, we might expect that the algorithm is helped by the fact that Linear is *fus*. Intuitively, a ruleset is *fus* if the rules can be compiled into the query by a simple backwards-chaining algorithm in a finite way[Roc16]. It is unlikely that this method could be extended to Guarded rules, as they are not *fus*.

3.5.2 Conjunctive Queries

Linear+*trans* for CQs is still an open question. Baget et al. showed that for binary rulesets, their algorithm would still terminate[Bag+15]. They go a step further and design a 'safety condition' that, when satisfied, guarantees termination even for non-binary rulesets. The safety condition ultimately guarantees that a sufficiently nice unifier μ always exists in the rewriting step such that any rewriting that would increase the size of the query is made redundant by rewritings using μ.

The safety condition is as follows:

- A predicate q is a *direct specialization* on a transitive predicate T on positions $(\vec{i},\vec{j})$ if neither $\vec{i}$ nor $\vec{j} = \emptyset$ and there is a rule of the form $q(\vec{u}) \longrightarrow T(x,y)$ where $\vec{i}$ contains exactly the positions in $\vec{u}$ where x occurs, and likewise for $\vec{j}$ and the positions where y occurs in $\vec{u}$.

  As an example, q in the rule $q(x,y,x,x,z) \longrightarrow T(x,y)$ is a direct specialization on $(\{1,3,4\},\{2\})$. If the rule were instead $q(x,y,x,x,z) \longrightarrow \exists w.T(w,y)$, then $\vec{i} = \emptyset$ and thus this rule would not cause q to be a specialization on T.

- A predicate q is a (not-necessarily-direct) *specialization* on T on $(\vec{i},\vec{j})$ if it is a direct specialization on $(\vec{i},\vec{j})$, or if there is a rule $q(\vec{u}) \longrightarrow p(\vec{w})$ where p is a specialization on T on positions $(\vec{\ell},\vec{m})$, where the terms in $\vec{\ell},\vec{m}$ in p occur in positions $\vec{i}$ and $\vec{j}$ in q, respectively.

  Using the first rule in the first example, and the rule $r(x,y) \longrightarrow \exists z.q(z,x,z,y,y)$, r would be a specialization on T on $(\{2\},\{1\})$.

- A ruleset is *safe* if, for every predicate q that specializes a transitive predicate, there exists a pair of positions (i,j) such that for all transitive predicates T upon which q specializes on $(\vec{i},\vec{j})$, either $i \in \vec{i}$ and $j \in \vec{j}$, or $j \in \vec{i}$ and $i \in \vec{j}$. *It is helpful to note that if all q specialize at most one transitive predicate, then the ruleset is trivially safe.*

  If q specializes on T on $(\{2\},\{1\})$ and $(\{3\},\{1\})$, the ruleset is not safe. However, if it specializes on T on $(\{2\},\{1\})$ and T on $(\{1\},\{2\})$, then the ruleset is safe.

The safety condition is notable in that it is entirely dependent on what can derive transitive atoms, rather than being dependent on what transitive atoms can derive. As we saw in Sections 3.2 and 3.4, many undecidability proofs rely on constructing a grid using transitivity for encoding a Turing machine over the grid to reduce to the halting problem. In these cases, the transitivity relations tend to appear near the start of the

ruleset (in terms of the derivation tree) - thus, there is not much space available for the safety condition to be violated, which implies that grid creation methods will not likely be fruitful in generating an undecidability result. The safety condition also allows us to achieve a CQ-decidability result for Linear+TG.

Theorem 1. *Linear+TG is decidable for CQs*

Proof. To see why this must be true, we need only note three things:

1. If a transitive atom appears in the body, it is a guard (since it is the only atom in the body).
2. If a transitive atom appears in the head, it is a guard if and only if it contains an existential variable (as it is the only atom in the head).
3. If a rule contains a transitive atom T in the head with an existential variable, then that rule does not create any specializations on T.

   To see this, note that any such rule is of a similar form to $p(x,...) \longrightarrow \exists y.T(x,y_\exists)$. $y_\exists$ never appears in p, so $\vec{j} = \emptyset$; but we had required that $\vec{j} \neq \emptyset$ for p to be a specialization on T.

Since there are no specializations on T, the safety condition is vacuously satisfied, which completes the proof. □

While this does show that Linear+TG is decidable, it is important to realize that the requirement that a Linear rule has an atomic head is often ignored, since one can always introduce a few auxiliary predicates to rewrite the ruleset into an atomic-headed one. However, this rewriting process will change which atoms serve as guards. Here is an example:

Original Ruleset:

$$p(x,y) \longrightarrow \exists z.T(y,z_\exists) \wedge q(x,z_\exists)$$

Atomic-Headed Equivalent Form:

$$p(x,y) \longrightarrow \exists z.\texttt{Auxiliary}(x,y,z_\exists)$$
$$\texttt{Auxiliary}(x,y,z) \longrightarrow T(y,z)$$
$$\texttt{Auxiliary}(x,y,z) \longrightarrow q(x,z)$$

It is natural to ask whether Linear+TG is still decidable if we relax the atomic head condition. The answer turns out to be yes.

Corollary 2. *Linear+TG is CQ-decidable, even if we do not require atomic heads.*

Proof. First, let us formally define the rewriting process that gives us an atomic-headed ruleset. For each rule $\mathcal{R}$, replace the head with an auxiliary predicate $\texttt{Aux}_\mathcal{R}$ which contains exactly once all the variables that appear in the head, and then for each atom h_i in the original head add the rule $\texttt{Aux}_\mathcal{R} \longrightarrow h_i$ (making sure to hook up the variables in the correct way so that the rulesets remain equivalent).

Now, we note that the only specialization of T is the auxiliary predicate, since the auxiliary predicate will always contain an existential in at least one of the two positions that specialize T (corresponding to the positions that used to be existential when T was a guard), so the specialization positions will be unable to propagate forwards.

Next, we note that each auxiliary predicate specializes only one transitive predicate. This is because every rule head can only contain one transitive predicate (or else one of them would not be the guard) - thus it would only be able to specialize a transitive predicate through another auxiliary predicate. As noted earlier, this is not possible.

We have shown that the only predicates which specialize a transitive predicate are auxiliary, and that these predicates specialize at most one transitive predicate. As noted when we defined the safety condition, this means that the ruleset must be safe. □

Chapter 4

Decidability Beyond the Guarded Fragment

This report so far has focused on the Guarded Fragment and its subclasses, but it is reasonable to wonder as to whether this is a natural place to focus. There are several generalizations of the concept of guardedness, such as weakly-guarded (WGTGDs), frontier-guarded (FGTGDs), and the Guarded Negation Fragment (GNF). In this section we will see that extending the transitivity results to these classes is in most cases a resolved question, thus showing that our focus on the Guarded Fragment is indeed reasonable. One exception too this is WGTGDs, where not much is yet known; this suggests an avenue for future research.

Each of these three classes use a more general notion of guardedness, and thus it is important for us to be clear on what it means for a ruleset to be "Base" and what it means for a ruleset to have transitivity only in guards (TG). For the definition of "Base", we can rely on the definition given by Amarilli et al. that a ruleset is Base$\mathcal{X}+$ *trans*, for some $\mathcal{X} \in \{\text{WGTGD}, \text{FGTGD}, \text{GNF}\}$, if transitivity atoms never appear in a weak-guard/frontier-guard/GNF-guard respectively[Ama+18]. In contrast, a ruleset is in $\mathcal{X}+\text{TG}$ if transitivity atoms only appear in guards *in the sense of the guarded fragment*. We have chosen these definitions as they are the most restrictive, and thus offer the most hope for decidability.

4.1 Guarded Negation Fragment

GNF	+Trans	+TG	Base+Trans
Atomic Queries	No[Grä98]	No	Yes[Ama+18]
CQs	No[Grä98]	No	Yes[Ama+18]
UCQs	No[Grä98]	No	Yes[Ama+18]

Table 4.1: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

The Guarded Negation Fragment is a powerful fragment which subsumes the Guarded Fragment. Like GF, GNF is most easily defined inductively:

1. All atoms are GNF
2. If ϕ, ψ are GNF, then $\phi \wedge \psi$, $\phi \vee \psi$, and $\exists \vec{z}.\phi(\vec{x}, \vec{z})$ are GNF.
3. If $\phi(\vec{x})$ is GNF and $A(\vec{x}, \vec{y})$ is an atom (each with no free variables other than explicitly noted), then $A(\vec{x}, \vec{y}) \wedge \neg\phi(\vec{x})$ is GNF.

As an example, $P(x,y) \wedge Q(x,y) \wedge (A(z,x) \wedge \neg\phi(z))$ is GNF, with A being a guard. However, $P(x,y) \wedge Q(x,y) \wedge (A(y,x) \wedge \neg\phi(z))$ is not GNF, as A no longer guards ϕ.

4.1.1 Atomic Queries

The only remaining question for atomic queries would be the compatibility of GNF with transitive guards. However, it is not entirely clear what a transitive guard is in this case. Let us recall that there are two conditions in GF which define a guard:

1. If there is a universal quantification, then it must be of the form $\forall z.A \longrightarrow \phi$, where ϕ is GF and A is an atom containing all the free variables of ϕ.
2. If there is an existential quantification, it must be of the form $\exists z.A \wedge \phi$, with the same conditions on A and ϕ.

In both cases, A is the 'guard'. The second condition is directly interpretable in GNF, but the first requires a bit more work. It is not hard to see, however, that it is logically equivalent to $\neg\exists z.A \wedge \neg\phi$, which can be interpreted in GNF. This gives us the following definition of GNF+TG: A formula in GNF is a formula in GNF+TG if all transitive predicates T appear in a subformula of one of the following forms (with T containing all the free variables of ϕ):

1. $\exists z.T \wedge \phi$
2. $\neg\exists z.T \wedge \neg\phi$

Theorem 2. *GNF+TG is atomic-undecidable.*

Proof. We can see that GNF+TG must be atomic-undecidable by taking a closer look at Gottlob et al's proof of GF+TG UCQ undecidability. During the proof, they constructed a sentence for which satisfiability was equivalent to the halting problem:

$$\hat{\varphi}_{grid} \wedge \varphi_M \wedge \neg(\hat{\varphi}_2 \vee \exists x.halt(x))$$

$$\hat{\varphi}_2 := \bigvee_{i,j} \exists xyx'y'.\gamma_{i,j} \wedge \psi_i \wedge H(x,y) \wedge V(x,x') \wedge V(y,y') \wedge \bar{H}(x',y')$$

$\gamma_{i,j}, \psi_i$ are conjunctions of atoms with variables x, y, x', y'

$\hat{\varphi}_{grid}, \varphi_M$ are GF+TG (and hence GNF+TG)

In their original proof, ψ_i is a conjunction of transitive atoms. However, if we add the rule $\forall xy.T(x,y) \longrightarrow T'(x,y)$ (equivalently $\neg\exists xy.T(x,y) \wedge \neg T'(x,y)$, which is GNF+TG) then we can replace all the T atoms in ψ_i with T' atoms without changing the satisfiability of the sentence. It should then be clear that $\hat{\varphi}_2$ is GNF+TG (since it is GNF and does not contain any transitive atoms). The sentence $\hat{\varphi}_2 \vee \exists x.halt(x)$ contains no free variables, so its negation is trivially GNF, and thus the whole sentence is GNF+TG.

As they had shown this sentence's satisfiability to be reducible to the halting problem, GNF+TG satisfiability, and hence atomic query answering, is undecidable. □

Corollary 3. *GNF+TG satisfiability is undecidable, as mentioned in Theorem 2.*

4.1.2 Unions of Conjunctive Queries

Amarilli et al. showed that BaseGNF+*trans* is UCQ decidable, an incredibly powerful result that shows as a corollary that Base$\mathcal{X}$+*trans* is UCQ decidable for nearly every class of rules considered in this report. Their proof is very technical, but it relies on showing that GNF, when equipped with a transitive closure operation, still satisfies the *bounded treewidth property*. By Courcelle's Theorem[Cou88], logical fragments that have the bounded treewidth property must admit a terminating algorithm for query answering. Courcelle's theorem does not actually tell us how to create such an algorithm, but Amarilli et al did manage to create an automaton that can answer UCQs[1].

[1]More precisely, the automaton determines BaseGNF+*trans* satisfiability, but UCQ answering for BaseGNF+*trans* can always be rephrased as a BaseGNF+*trans* satisfiability problem.

4.2 Frontier-Guarded

Frontier Guarded Rules	+Trans	+TG	Base+Trans
Atomic Queries	No[GPT13]	No	Yes[Ama+18]
CQs	No[GPT13]	No	Yes[Ama+18]
UCQs	No[GPT13]	No	Yes[Ama+18]

Table 4.2: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

We say that a ruleset is frontier-guarded if, for every rule, there is an atom in the body which contains all of the frontier variables. FGTGDs can be shown to be a subclass of GNF; the UCQ decidability of BaseFGTGD+*trans* is thus inherited from that of BaseGNF[Ama+18]. Likewise, they are a generalization of GTGDs, so the atomic undecidability of FGTGD+*trans* follows as well. Thus the only unknown is the decidability of FGTGD+TG. This is in fact undecidable, even for atomic query answering.

Theorem 3. *FGTGD+TG is atomic-undecidable.*

Proof. First, observe that FGTGD+TG = FGTGD+TFG, where "TFG" stands for 'transitivity only in frontier-guards'[2]. To see why, consider that we can add a FGTGD+TG rule $T(x,y) \longrightarrow T'(x,y)$ to any FGTGD+TFG ruleset, and then replace all instances of T in frontier guards with T'. This makes the entire ruleset FGTGD+TG.

We can use FGTGD+TFG to reduce to the known-to-be-undecidable problem of checking whether the intersection of two context free languages contains the same word. This is heavily inspired by a result by Krötzsch and Rudolph[KR], who used the following ruleset as a demonstration that their proposed 'Directional Rules' ruleclass was undecidable for CQs (for simplicity, it has been edited to assume that the grammar is in Chomsky Normal Form and edited so that it accepts an atomic, rather than conjunctive, query):

$$\bigwedge_{\text{terminals and nonterminals } \alpha \text{ in the context free grammar (CFG)}} \forall x.\exists y.r_\alpha(x,y)$$

$$\bigwedge_{\text{grammar rules of the form } \alpha{:=}\beta\gamma} r_\beta(x,y) \wedge r_\gamma(y,z) \longrightarrow r_\alpha(x,z)$$

$$\bigwedge_{\text{grammar rules of the form } \alpha{:=}a} r_a(x,y) \longrightarrow r_\alpha(x,y)$$

$$r_{S_1}(x,y) \wedge r_{S_2}(x,y) \longrightarrow \texttt{NonemptyIntersection}(x)$$

Assuming the two grammars had start symbols S_1, S_2, this will check if the grammars both generate a common string. While the first rule is not directly allowed in

[2]Specifically, we will require that transitive predicates in the body appear in the frontier guard, and that transitive predicates in the head contain all existential variables as usual. We do not require that head predicates contain all frontier variables.

Datalog$^{\pm}$, it is not hard to mimic it using linear rules; for every predicate p, add the rule $p(x,...) \longrightarrow \exists y.r_\alpha(x,y)$ (repeat for each possible position of x in p).

This ruleset is not FGTGD, but we can make it so by adding a transitive predicate T:

$$\bigwedge_{\text{terminals and nonterminals } \alpha \text{ in the context free grammar (CFG)}} \forall x.\exists y.r_\alpha(x,y) \wedge T(x,y)$$

$$\bigwedge_{\text{grammar rules of the form } \alpha := \beta\gamma} r_\beta(x,y) \wedge r_\gamma(y,z) \wedge T(x,z) \longrightarrow r_\alpha(x,z)$$

$$\bigwedge_{\text{grammar rules of the form } \alpha := a} r_a(x,y) \longrightarrow r_\alpha(x,y)$$

$$r_{S_1}(x,y) \wedge r_{S_2}(x,y) \longrightarrow \texttt{NonemptyIntersection}(x)$$

It is not hard to see that T only appears in frontier guards, so this ruleset is an instance of FGTGD+TFG. `NonemptyIntersection` is entailed if and only if the languages share a string, so atomic query entailment is undecidable for FGTGD+TFG, and hence for FGTGD+TG as well. □

4.3 Weakly-Guarded

Weakly Guarded Rules	+Trans	+TG	Base+Trans
Atomic Queries	No[GPT13]	?	?
CQs	No[GPT13]	?	?
UCQs	No[GPT13]	?	?

Table 4.3: Results in grey follow trivially from other results in the report; uncited results are results original to this report.

We say that a ruleset is weakly-guarded if, for every rule, there is an atom in the body which contains all of the 'affected' variables, where affectedness is defined in terms of the following condition on the positions of a predicate[CGK13]:

1. If in the head of some rule an existential variable occurs in position $p[i]$, we mark $p[i]$ as 'affected'.
2. If in the head of some rule a frontier variable occurs in position $p[i]$, and the positions in which this frontier variable occurs in the body are all affected positions, we mark $p[i]$ as affected.
3. A body variable is affected if it appears exclusively in affected positions within the body.

Like FGTGDs, WGTGD+*trans* inherits its atomic undecidability from GTGDs; however, WGTGD is not a subclass of GNF - we know nothing a priori about the decidabilities of BaseWGTGD+*trans* or WGTGD+TG.

It is likely to be the case that the (un)decidabilities are the same as those for GTGDs; there are several rewriting methods which link the expressive powers of GTGDs, WGTGDs, and their generalizations. For example, WGTGDs can be rewritten as GTGDs[Bag+11], and FGTGDs can be rewritten as WGTGDs[GRS14]. Additionally, if an atom appears only in guards, these rewritings tend to preserve that fact.

Unfortunately, the rewritings do not preserve the transitivity rule, which is what prevents us from concluding that WGTGD+TG is equivalent to GTGD+TG in terms of expressive power. Thus, it is possible that WGTGD+TG and GTGD+TG are fundamentally different. There is precedent for this; binary cross products (rules of the form $C_1(x) \wedge C_2(y) \longrightarrow \bar{C}(x,y)$) are known to be compatible with GTGDs[BMP17] but not with WGTGDs[3]. However, the incompatibility proof relies on the fact that cross products have different head and body predicates, which allows them to isolate affected positions and cause every rule to be trivially weakly guarded. Transitivity, on the other hand, tends to respect the weakness condition - in fact, it is not hard to see that the transitivity rule can never change which positions are affected.

[3]The proof of this is given in Appendix A.3

Chapter 5

Complexity Results

In this chapter, we will aim to gain an understanding of the precise computational complexity of the positive decidability results obtained in previous chapters. These complexity results are compiled into Table 5.1.

Complexity Results	Data Complexity			Combined Complexity		
Query Language	Atomic	CQ	UCQ	Atomic	CQ	UCQ
Linear	in AC_0[a]	in AC_0[a]	?	PSpace-c[b]	PSpace-c[c]	?
Disjunctive Linear	in Logspace[a]	coNP-c[a]	coNP-c[d]	Exp-c[e]	2Exp-c[d]	2Exp-c[d]
Guarded	PTime-c[a]	PTime-c[a]	?	2Exp-c[e]	2Exp-c[c]	2Exp-c
Disjunctive Guarded	coNP-c[a]	coNP-c[a]	coNP-c[d]	2Exp-c[e]	2Exp-c[d]	2Exp-c[d]
Frontier Guarded	PTime-c	PTime-c[c]	?	2Exp-c	2Exp-c[c]	2Exp-c
Guarded Fragment	coNP-c	coNP-c	coNP-c	2Exp-c	2Exp-c[f]	2Exp-c[f]
Guarded Negation Fragment	coNP-c[g]	coNP-c[g]	coNP-c[g]	2Exp-c[g]	2Exp-c[g]	2Exp-c[g]
Linear+*trans*	NL-c[b]	?	?	Exp-c[b]	?	?
Linear+TG	NL-c	NL-c	?	in Exp	in Exp	?
BaseLinear+*trans*	NL-c	?	?	Exp-c	?	?
BaseD-Linear+*trans*	?	coNP-c	coNP-c	?	2Exp-c	2Exp-c
BaseGuarded+*trans*	?	?	?	2Exp-c	2Exp-c	2Exp-c
BaseD-Guarded+*trans*	coNP-c	coNP-c	coNP-c	2Exp-c	2Exp-c	2Exp-c
BaseFrontier-Guarded+*trans*	?	?	?	2Exp-c[g]	2Exp-c[g]	2Exp-c[g]
BaseGF+*trans*	coNP-c	coNP-c	coNP-c	2Exp-c	2Exp-c	2Exp-c
BaseGNF+*trans*	coNP-c[g]	coNP-c[g]	coNP-c[g]	2Exp-c[g]	2Exp-c[g]	2Exp-c[g]

Table 5.1: For reference, the complexities for the considered classes without transitivity are also given. A suffix of -c indicates completeness, -h indicates hardness. Grey cells indicate that decidability is not known. Uncited results are those which follow from cited results.

[a] [Alv+12]
[b] [Bag+15]
[c] [Mug11]
[d] [BMP13]
[e] [Got+12]
[f] [BGO10]
[g] [Ama+18]

5.1 Linear Rules with Transitivity and Transitive Guards

Baget et al. provided complexity results along with their decidability results[Bag+15]. They established that:

1. Atomic query answering for Linear+*trans* is NL-complete in data complexity and Exptime-complete in combined complexity.
2. CQ entailment for safe Linear+*trans* is NL-complete in data complexity, and it is in Exptime for combined complexity (Exptime-completeness is not known).

They achieved upper bounds on complexity through an analysis of the algorithm they presented to demonstrate decidability. Combined complexity lower bounds were achieved through a modification of a proof by Bienvenue and Thomazo[BT16], by simulating arbitrary PSpace alternating Turing machines (ATM) and noting that the problem of checking whether a word was generated by a given PSpace ATM is Exptime-complete. Data complexity lower bounds were achieved from a reduction from the NL-complete directed reachability problem (i.e. given a directed graph, is one vertex reachable from the other?).

NL-completeness is in fact inherent to transititivity, not Linear rules. Suppose we only had a single transitive predicate with no Linear rules, and some set of initial facts consisting of transitive atoms. Each atom can be thought of as defining a directed edge of a graph, and it should be clear that the concept of reachability is encoded by the transitive relation. As the directed reachability problem is NL-complete, atomic Linear+*trans* must be NL-hard (and hence NL-complete, as Baget et al's algorithm has complexity NL). This is a very nice result, as it shows that the complexity of Linear+*trans* is the best result we could have hoped for.

Remark 2. *Linear+TG is NL-complete in data complexity for CQ query answering.*

Proof. In this report we saw a reduction from Linear+TG to safe Linear+*trans*, so Linear+TG inherits the upper bounds for safe Linear+*trans*. Furthermore, Linear+TG must be NL-complete in data complexity for conjunctive query answering since the same reduction to directed reachability as in the last paragraph still holds. □

For BaseLinear+*trans* on atomic queries, we inherit NL and Exptime upper bounds for data and combined complexity from the general Linear+*trans* case, and we can see that it must indeed be NL-complete as well by the same reasoning as the previous remark.

Remark 3. *BaseLinear+trans is ExpTime-complete in combined complexity for atomic query answering.*

Proof. We can achieve an Exptime-complete combined complexity for BaseLinear+*trans*. As mentioned, Baget et al. proved Exptime-completeness for general Linear+*trans* by reducing from the word acceptance problem for PSpace alternating Turing machines. Importantly their Linear+*trans* ruleset used in the reduction never uses a transitive predicate in the body; and hence it is a BaseLinear+*trans* ruleset as well. □

5.2 “Base-” Rules

Amarilli et al. showed that transitivity can be added to BaseGNF with resultant combined complexity 2Exptime-complete and data complexity coNP-complete[Ama+18]; this is in fact the best possible result, as GNF is 2Exptime-complete even without transitivity. From this result we immediately gain a near-complete understanding for the combined complexity landscape of all subclasses of GNF, however the picture for data complexity is not so rosy. They do note that all data complexities are coNP-complete in the case of transitive **closure** predicates (even with BaseLinear rules), but this is due to the fact that transitive closure on its own has coNP-complete data complexity (analogously to how transitivity on its own has NL-complete data complexity) - thus their completeness proof is likely not extensible to the case of transitivity.

Chapter 6

Conclusion

In this report we have seen that the picture of transitive compatibility with guarded rules is clearing up. Despite this, there are still many open questions with (D-)Linear rules and the state of GTGDs+TG is largely unknown as well. Most natural ways to generalize the Guarded Fragment and its subclasses, such as GNF and FGTGDs, have also had the question of transitive compatibility resolved, although we do not know much about the compatibility of transitivity with WGTGDs.

One possible avenue for future work would be to clear up the open questions outlined in the previous paragraph - of particular interest is the case of Linear rules. Unlike general transitivity and transitive guards, no undecidability results have been proven for Base- rulesets. A natural question is how complicated can a ruleclass $\mathcal{X}$ get before Base$\mathcal{X} + \mathit{trans}$ becomes undecidable; the first step towards this would be to investigate weakly-guarded rules and their generalizations. Yet another avenue for improvement would be to begin work towards filling in the missing complexity values of Table 5.1; most work so far has been dedicated to establishing decidability, with complexity being an afterthought.

While there are still open questions, this survey represents a complete overview of the currently known compatibility results for guarded rules and transitivity. We have seen that the question of compatibility is now closed for GNF, GF, GTGDs, and FGTGDs, with the help of some original results. The question of general compatibility with transitive rules is open only for Linear UCQ answering and D-Linear atomic answering, whereas compatibility with transitivity only in guards still has quite a few open questions left. Most cited papers were written in the last ten years; if this rate keeps up, it is not unreasonable to expect that the problems posed in this survey will be resolved within the next decade.

Bibliography

[Cou88] Bruno Courcelle. “The monadic second-order logic of graphs, II: Infinite graphs of bounded width”. In: *Mathematical Systems Theory* 21.1 (1988), pp. 187–221.

[ANB98] Hajnal Andréka, István Németi, and Johan van Benthem. “Modal languages and bounded fragments of predicate logic”. In: *Journal of philosophical logic* 27.3 (1998), pp. 217–274.

[Grä98] Erich Grädel. “On the Restraining Power of Guards”. In: *Journal of Symbolic Logic* 64 (1998), pp. 1719–1742.

[ST01] W. Szwast and L. Tendera. “On the decision problem for the guarded fragment with transitivity”. In: *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*. 2001, pp. 147–156. DOI: 10.1109/LICS.2001.932491.

[BGO10] Vince Bárány, Georg Gottlob, and Martin Otto. “Querying the guarded fragment”. In: *2010 25th Annual IEEE Symposium on Logic in Computer Science*. IEEE. 2010, pp. 1–10.

[Bag+11] Jean-François Baget et al. “Walking the Complexity Lines for Generalized Guarded Existential Rules”. In: *IJCAI: International Joint Conference on Artificial Intelligence*. Ed. by Toby Walsh. Barcelona, Spain: AAAI Press, July 2011, pp. 712–717. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-00618081.

[Mug11] Marie-Laure Mugnier. “Ontological query answering with existential rules”. In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2011, pp. 2–23.

[Alv+12] Mario Alviano et al. “Disjunctive datalog with existential quantifiers: Semantics, decidability, and complexity issues”. In: *arXiv preprint arXiv:1210.2316* (2012).

[Got+12] Georg Gottlob et al. “On the complexity of ontological reasoning under disjunctive existential rules”. In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 2012, pp. 1–18.

[BMP13] Pierre Bourhis, Michael Morak, and Andreas Pieris. “The impact of disjunction on query answering under guarded-based existential rules”. In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[CGK13] A. Calì, G. Gottlob, and M. Kifer. “Taming the Infinite Chase: Query Answering under Expressive Relational Constraints”. In: *Journal of Artificial Intelligence Research* 48 (Oct. 2013), pp. 115–174. ISSN: 1076-

9757. DOI: 10.1613/jair.3873. URL: http://dx.doi.org/10.1613/jair.3873.

[GPT13] Georg Gottlob, Andreas Pieris, and Lidia Tendera. "Querying the Guarded Fragment with Transitivity". In: *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part II*. ICALP'13. Riga, Latvia: Springer-Verlag, 2013, pp. 287–298. ISBN: 9783642392115. DOI: 10.1007/978-3-642-39212-2_27. URL: https://doi.org/10.1007/978-3-642-39212-2_27.

[GRS14] Georg Gottlob, Sebastian Rudolph, and Mantas Simkus. "Expressiveness of guarded existential rule languages". In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2014, pp. 27–38.

[Bag+15] Jean-François Baget et al. "Combining Existential Rules and Transitivity: Next Steps". In: *CoRR* abs/1504.07443 (2015). arXiv: 1504.07443. URL: http://arxiv.org/abs/1504.07443.

[BT16] Meghyn Bienvenu and Michaël Thomazo. "On the complexity of evaluating regular path queries over linear existential rules". In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2016, pp. 1–17.

[Roc16] Swan Rocher. "Querying existential rule knowledge bases: decidability and complexity". PhD thesis. Université Montpellier, 2016.

[BMP17] Pierre Bourhis, Michael Morak, and Andreas Pieris. "Making cross products and guarded ontology languages compatible". In: *IJCAI 2017-Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2017, pp. 880–886.

[Ama+18] Antoine Amarilli et al. "Query Answering with Transitive and Linear-Ordered Data". In: *Journal of Artificial Intelligence Research* 63 (Sept. 2018), pp. 191–264. DOI: 10.1613/jair.1.11240. URL: https://hal.telecom-paris.fr/hal-02338348.

[KR] Markus Krötzsch and Sebastian Rudolph. "Directional Rules: Tractable Datalog+/–with Transitivity". In: ().

Appendix A

Complementary Proofs

A.1 Cross Products

A binary cross product is a rule of the form $C_1(x) \wedge C_2(y) \longrightarrow \bar{C}(x,y)$, and will often be abbreviated as $\times_2$. They are known to be compatible with guarded rules[BMP17], and hence they are an interesting case study to compare with the compatibility of transitivity rules.

A.2 D-Linear $+ \times_2$ is CQ undecidable

As mentioned in Section 3.1, the following ruleset encodes a known-to-be-undecidable tiling problem.

$$
\begin{aligned}
S'(x,y) &\longrightarrow \exists z.S'(y,z_\exists) \\
S'(x,y) &\longrightarrow S^+(x,y) \\
S^+(x,y) &\longrightarrow \bigvee_i K_i(x,y) \\
S^+(x,y) \wedge S^+(y,z) &\longrightarrow S^+(x,z) \text{ (transitivity)} \\
S^+(x,y) &\longrightarrow S^+(y,x) \text{ (symmetry)} \\
\bigwedge_{0 \le i < n} S'(a_i,a_{i+1}) \wedge & \bigwedge_{0 \le i \le n} K_j(a_0,a_i) \text{ (facts)}
\end{aligned}
$$

There is a single transitive predicate S^+, which satisfies a "connectedness" property, defined in terms of a graph. This graph will have its vertices be terms, and there will be an edge between x and y if $S^+(x,y)$ is true. As $S+$ is symmetric, this graph is undirected. The connectedness condition is simply that this graph (which is related to the Gaifman graph) is connected. It is not hard to convince yourself that this must be true - all the initial S' facts lead to a connected set of S^+ facts, and the first and second rules of the ruleset preserve this connectedness.

Note that if some term x is a vertex on this graph (i.e. if it appears in a transitive atom), then it is connected to all other vertices on this graph (i.e. $S^+(x,y)$ is true for all y that appear in transitive atoms). Thus the property of transitivity can be replaced with a cross product;

$$\begin{aligned} S'(x,y) &\longrightarrow \exists z.S'(y,z_\exists) \\ S'(x,y) &\longrightarrow C(x) \wedge C(y) \\ S^+(x,y) &\longrightarrow \bigvee_i K_i(x,y) \\ C(x) \wedge C(y) &\longrightarrow S^+(x,z) \text{ (cross product)} \\ \bigwedge_{0 \le i < n} S'(a_i,a_{i+1}) \wedge \bigwedge_{0 \le i \le n} K_j(a_0,a_i) &\text{ (facts)} \end{aligned}$$

These rulesets are equivalent, which shows that D-Linear+$\times_2$ is UCQ undecidable. In fact, Amarilli et al's UCQ to CQ conversion method still applies, and additionally this ruleset uses only disjunctive inclusion dependencies, leading to the stronger result that DID+$\times_2$ is CQ undecidable.

A.3 WGTGD $+ \times_2$ is atomic undecidable

We note that cross products give us a very convenient property; the predicates in the head of a cross product rule are not the same as those in the body. We can use this to effectively destroy classes of rules that are based around keeping track of the positions of variables, as it allows variables to 'teleport'.

Let's consider arbitrary TGDs with atomic heads and at most one existential variable. Atomic query answering is undecidable for this problem, as completely arbitrary TGDs can all be written in this form. For every rule $\vec{B}(\vec{x},\vec{y}) \longrightarrow \exists z.H(\vec{x},z)$, we can turn it into two rules:

$$\begin{aligned} \vec{B}(\vec{x},\vec{y}) &\longrightarrow \exists z.\bar{H}(\vec{x}) \wedge C(z) \\ \bar{C}(x,x) \wedge \bar{H}(\vec{y}) &\longrightarrow H(\vec{y},x) \\ (C(x) \wedge C(y) &\longrightarrow \bar{C}(x,y)) \text{ [Cross Product]} \end{aligned}$$

Note that the only affected position is $C[1]$, but it never appears in the body of a rule (except the cross product rule). Thus all rules are trivially guarded. This proves the reduction from arbitrary TGDs to $WGTGD + \times_2$, so the latter is not decidable for atomic query answering.