Rule-aware Datalog Fact Explanation Using Group-SAT Solver

Akira Charoensit[1], David Carral[1], Pierre Bisquert[1,2], Lucas Rouquette[1] and Federico Ulliana[1]

[1]Inria, LIRMM, Univ Montpellier, CNRS, France
[2]IATE, Univ Montpellier, INRAE, Institut Agro, Montpellier, France

Abstract

One of the major benefits of symbolic AI is explainability. When new knowledge is obtained via a reasoning process, it is possible to determine precisely all elements of the knowledge base that yield this knowledge. Typically, one would use a SAT solver to compute the explanations. However, SAT-solving is computationally expensive, and as the knowledge base grows, the time required increases exponentially. This work presents a method for filtering a datalog knowledge base to optimise the time used by a SAT solver. This is achieved by creating a hypergraph representing the grounded knowledge base and pruning the nodes that are not reachable from the fact that we want to explain. This approach proves to be time-effective. Interestingly, one additional benefit of using this hypergraph is that it is possible to encode more information about the rules used in the reasoning process. By using an off-the-shelf group-SAT solver, this extra information allows us to find specific explanations that would be missed if we only considered facts.

1. The Explanation Issue

In the realm of Artificial Intelligence, symbolic AI stands out for its ability to provide explainable results and predictions. In particular, knowledge and rule-based systems make it possible for users to trace the derivation of new knowledge back to the exact elements that have contributed to it. This traceability is crucial for applications where understanding the reasoning process is as important as the outcome itself, such as in healthcare or regulatory enforcement.

While the issue of computing explanations has long been studied for Description Logics (see e.g., [1, 2, 3, 4, 5]), the extension of the approach to mainstream rule languages such as Datalog has not been considered so far. Datalog is widely recognised as a language which leverages recursion for data processing [6] and plays a significant role in ontology-mediated query answering [7]. On the one hand, it encompasses RDF-Schema and OWL-RL ontologies. On the other hand, many reasoning tasks on ontologies can be reduced to query answering in this language [8]. Providing explanations for Datalog is thus a step towards more reliable data-intensive applications, and the goal of this work is to contribute to the explainability of Datalog.

Explaining reasoning in Datalog raises, however, two key challenges. The first is to choose a form of explanation that is suitable for the reasoning task. The second is to actually compute them, which is known to be expensive even for basic forms of explanations [9, 10].

Choosing Explanations Let us illustrate the importance of choosing the "right" notion of explanation. Hereafter, we assume the reader is familiar with propositional and first order logic. Consider the knowledge base $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ in Figure 1 with facts $\mathcal{F} = \{\mathsf{Boss}(\mathsf{alice}, \mathsf{alice})\}$ and rules $\mathcal{R} = \{r_1, r_2, r_3\}$. Consider the task of explaining the fact $\varphi = \mathsf{Manager}(\mathsf{alice})$, which is entailed by $\mathcal{K}$. As Figure 1 illustrates, φ can be derived in two different ways starting from $\mathcal{F}$. One is by applying r_1. The other is by applying r_2 and then r_3.

RuleML+RR'24: Companion Proceedings of the 8th International Joint Conference on Rules and Reasoning, September 16–22, 2024, Bucharest, Romania

akira.charoensit@inria.fr (A. Charoensit); david.carral@inria.fr (D. Carral); pierre.bisquert@inrae.fr (P. Bisquert); lucas.rouquette@inria.fr (L. Rouquette); federico.ulliana@inria.fr (F. Ulliana)

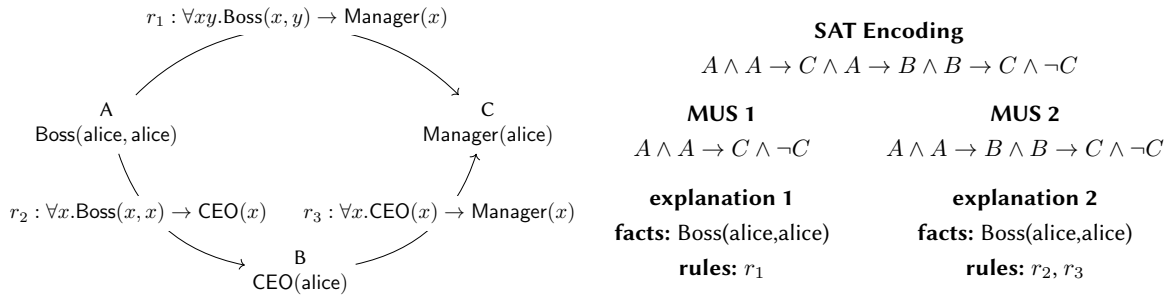

Figure 1: Computing Minimal KB-Support Explanations via MUSes

One possible form of explanation consists in computing the *fact-support* of φ with respect to $\mathcal{F}$, that is, the set of facts $\mathcal{F}' \subseteq \mathcal{F}$ that can contribute to the derivation of φ. The notion of fact-support corresponds to that of *why-provenance* [8, 10, 9, 5]. For the example in Figure 1, this would yield $\mathcal{F}' = \mathcal{F}$. Here, fact-support provides an insufficient information as it does not show the role of rules in the reasoning task.

A meaningful notion of explanation here would be that of *kb-support*, that is, a subset $\mathcal{K}'$ of $\mathcal{K}$ which entails φ. To be precise, we need to identify *minimal subsets of the knowledge base that preserve the entailment* since users typically prefer *concise* explanations. In the example of Figure 1, there are two kb-support explanations: $\mathcal{K}'_1 = \langle\{r_1\}, \mathcal{F}\rangle$ and $\mathcal{K}'_2 = \langle\{r_2, r_3\}, \mathcal{F}\rangle$. The interest in the notion of kb-support is that it better explains the roles taken by both the rules and the data in the reasoning task thereby giving the user more power to potentially take action and revise the knowledge base. In the context of Description Logics, kb-supports correspond to the notion of *justifications with ABoxes* (the standard version of justifications rather considering entailment axioms over TBoxes) [1, 2]. In this work, we will thus consider this notion of kb-support which enables more detailed explanations for fact entailment over Datalog knowledge bases.

Computing Explanations An effective approach to tackling the expensive theoretical cost of computing explanations is to rely on the use of SAT-solvers [2, 8]. Nowadays, these are considered mature and effective tools. The critical step of this reduction is that of *encoding the input knowledge base and entailment* (i.e., the inputs of the explanation problem) *as a propositional formula.* To illustrate, Figure 1 shows the SAT encoding of the explanation problem, which goes as follows. It associates every atom entailed by the knowledge base with a distinct propositional variable (here, A, B, and C). Then, a conjunctive formula is built. Its elements are either:

(i) a propositional variable corresponding to a fact in $\mathcal{F}$ (e.g., A represents $\mathsf{Boss}(\mathsf{alice}, \mathsf{alice})$),

(ii) a grounded rule corresponding to a rule application (e.g., $A \rightarrow B$ represents $\mathsf{Boss}(\mathsf{alice}, \mathsf{alice}) \rightarrow \mathsf{CEO}(\mathsf{alice})$) or

(iii) a negated literal, whose propositional variable corresponds to the entailment (e.g., $\neg C$ corresponds to $\neg\mathsf{Manager}(\mathsf{alice})$).

An important detail here is that negating the fact to explain in the encoded formula allows one to solve the explanation problem by looking at its *minimal unsatisfiable sets* (MUSes) [11]. As Figure 1 shows, each MUS of the encoded formula provides an explanation.

Computing *All* Explanations Computing explanations via MUS for Datalog can lead to potential pitfalls. The main issue is that two distinct rule applications can "conflict" by resulting in the same SAT-encoding. This case is illustrated in Figure 2. Here, we can see that the applications of r_1 and r_2 are both encoded as $A \rightarrow C$. As a result, the MUS enumeration no longer yields a (minimal) explanation. The situation becomes even more complex when many recursive rules share the same groundings. To

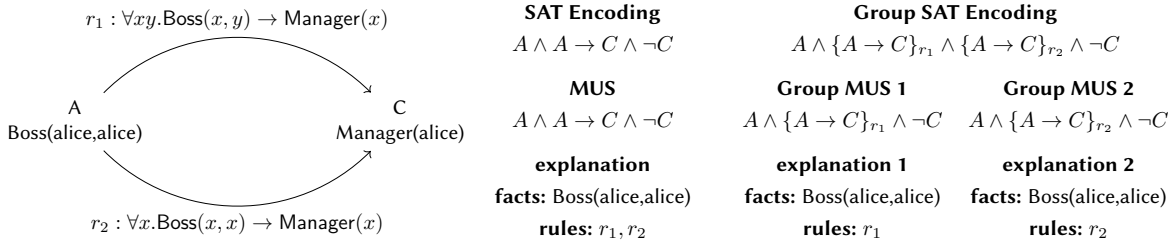

Figure 2: SAT vs Group-SAT Translation Example

address these issues, we present an encoding into *group*-SAT formulas [11] which allows us to establish a precise correspondence between explanations and *group*-MUSes.

Computing All Explanations *Efficiently* While solvers are efficient, MUS computation remains hard. Therefore, significant challenges may arise in terms of scalability as the size and complexity of the knowledge base increase. The main reason for this is that in more complex knowledge bases there may be a larger number of atoms and rules which are *not relevant for the entailment to explain*. This is illustrated in Figure 3: atoms $\mathsf{s}(\mathsf{c})$ and $\mathsf{s}(\mathsf{d})$, as well as rule r_4, are irrelevant for explaining $\mathsf{goal}(\mathsf{a})$. Considering irrelevant atoms will slow down both the encoding computation and the MUS enumeration task. Hence, we introduce a filtering technique that reduces the size of the encoded formula while preserving its soundness and completeness.

Contribution The contributions of this paper are the following:

1. We introduce the first comprehensive approach for computing kb-support explanations for fact entailment (a.k.a justifications with ABoxes) over Datalog knowledge bases.
2. We present a reduction from Datalog rules and facts to group-SAT formulas, and establish an exact correspondence between kb-support explanations and group-MUSes (Section 2).
3. We study the filtering of facts that are irrelevant for the explanation task and show that it is computationally hard. In light of this, we present a two-step rule-based approach for approximating relevance; this includes a static step preprocessing of the input KB and a dynamic step for tracing a single entailment. (Section 3).
4. We present an experimental evaluation showing the effectiveness of our approach. In particular, we show that the two-step approach allows good performances to dynamically explain any entailment query (Section 4).

2. From Datalog Explanations to Group-MUS

The Logical Setting We assume a first-order signature with functions. We consider mutually disjoint sets of variables, constants, and functions. We call *term* any variable, constant, or functional term of the form $f(t_1, \ldots, t_n)$ where f is a function symbol and every t_i is a term. We write lists $t_1, \ldots, t_n$ of terms as $\bar{t}$ and often treat these as sets. An *atom* is a first-order formula of the form $P(\bar{t})$ where P is a relational predicate and $\bar{t}$ a sequence of terms. An atom is *grounded* if it does not contain any variable. For a first-order formula Φ and a list $\bar{x}$ of variables, we write $\Phi[\bar{x}]$ to indicate that $\bar{x}$ is the set of all free variables that occur in Φ. A *fact* is a grounded atom without function symbols. A *rule* r is a first-order formula of the form $\forall \bar{x}.B[\bar{x}] \rightarrow H[\bar{z}]$ where $B[\bar{x}]$ and $H[\bar{z}]$ are conjunctions of atoms and $\bar{z} \subseteq \bar{x}$. Such a rule is called *Datalog* if it is function-free and H is a single atom. We will often omit universal quantifiers when writing rules. Moreover, we identify conjunctions of atoms such as B and H above with the corresponding sets, and define $\mathtt{body}(r) = B$ and $\mathtt{head}(r) = H$. A *knowledge*

base (KB) $\mathcal{K}$ is a tuple $\langle \mathcal{R}, \mathcal{F} \rangle$ where $\mathcal{R}$ and $\mathcal{F}$ are finite sets of rules and facts, respectively. A *Datalog knowledge base* is such that $\mathcal{F}$ only contains grounded atoms without function symbols and $\mathcal{R}$ only contains Datalog rules. For $\mathcal{K}_1 = \langle \mathcal{R}_1, \mathcal{F}_1 \rangle$ and $\mathcal{K}_2 = \langle \mathcal{R}_2, \mathcal{F}_2 \rangle$ we write $\mathcal{K}_1 \subseteq \mathcal{K}_2$ when $\mathcal{R}_1 \subseteq \mathcal{R}_2$ and $\mathcal{F}_1 \subseteq \mathcal{F}_2$.

Fact Entailment The *chase* is a standard method for computing universal models of knowledge bases that can in turn be used for tasks like fact entailment [12]. A *substitution* is a partial function mapping variables to ground terms. A *trigger* for a fact set $\mathcal{F}$ and a rule set $\mathcal{R}$ is a tuple $\tau = \langle r, \sigma \rangle$ where $r \in \mathcal{R}$, and σ is a substitution such that $\sigma(\texttt{body}(r)) \subseteq \mathcal{F}$. The trigger outputs $\texttt{out}(\tau) = \sigma(\texttt{head}(r))$. For a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$, let $\texttt{Chase}_0(\mathcal{K}) = \mathcal{F}$; and, for every $i \geq 1$, let $\texttt{Chase}_i(\mathcal{K})$ be the minimal fact set that includes $\texttt{Chase}_{i-1}(\mathcal{K})$ and $\texttt{out}(\tau)$ for every trigger τ of $\texttt{Chase}_{i-1}(\mathcal{K})$ and $\mathcal{R}$. Moreover, let $\texttt{Chase}(\mathcal{K}) = \bigcup_{i \geq 0} \texttt{Chase}_i(\mathcal{K})$. A KB $\mathcal{K}$ entails a fact φ, denoted $\mathcal{K} \models \varphi$, if and only if $\varphi \in \texttt{Chase}(\mathcal{K})$.

From now on, every knowledge base considered in this section is *Datalog*. Knowledge bases with functions will be employed in Section 3. At this point, we can formally define the notion of explanation for Datalog knowledge bases.

Definition 1 (Explanation)**.** *For a Datalog knowledge base $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ and a fact φ, a kb-support* explanation *of $\mathcal{K} \models \varphi$ is a KB $\mathcal{K}' \subseteq \mathcal{K}$ such that $\mathcal{K}' \models \varphi$ but $\mathcal{K}'' \not\models \varphi$ for every $\mathcal{K}'' \subset \mathcal{K}'$. We denote by $\texttt{Expl}(\mathcal{K}, \varphi)$ the set of all explanations of $\mathcal{K} \models \varphi$.*

From Explanations to Group-MUS We now show how to reduce the explanation problem to group-MUS (GMUS) enumeration. Group-SAT (GSAT) formulas are natural extensions of SAT formulas where constraints are modeled as sets or groups [13]. The underlying idea is that all clauses in a group must hold together. Below, we assume the standard notion of literal (positive or negative propositional variable) and clause (disjunction of literals). A *group* $\mathcal{G}$ is a *set* of clauses. To keep the formalisation concise, we slightly enrich the standard definition and also assume that *every group $\mathcal{G}$ has a unique identifier* i denoted by $\mathcal{G}_i$. Otherwise said, we consider two groups with the same clauses but with different identifiers to be different. A *GSAT* formula $\mathfrak{F}$ is a set of groups $\mathfrak{F} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$. A GSAT formula is *satisfiable* if the conjunction of all the clauses in the union of its groups is satisfiable. A *GMUS* of $\mathfrak{F}$ is a *minimal* set of the groups of $\mathfrak{F}$ that is *unsatisfiable*. We write $\texttt{GMUS}(\mathfrak{F})$ for the set of all GMUS of $\mathfrak{F}$. Figure 2 illustrates the GSAT formula stemming from the encoding of the input facts and rules. In the formula, we have two groups which contain the same (set of) clauses; these are denoted as $\{A \to C\}_{r_1}$ and $\{A \to C\}_{r_2}$ where the identifier of each group corresponds to the rule that generates it. Also note that with this encoding every GMUS in Figure 2 corresponds to one explanation.

To reduce the computation of the explanations for a fact ψ over a KB $\mathcal{K}$ to group-MUS enumeration, our goal is to produce a GSAT formula which encodes the derivations enabled by the KB as well as the (negation of) the fact to explain. Let $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$, we define $\texttt{GSAT}(\mathcal{K} \wedge \neg\psi)$ as the minimal set containing:

- the group $\mathcal{G}_r$ with $\mathcal{G} = \texttt{Grounding}_{\mathcal{K}}(r)$, for every $r \in \mathcal{R}$
- the group $\mathcal{G}_\varphi$ with $\mathcal{G} = \{\varphi\}$, for every $\varphi \in \mathcal{F}$
- the group $\mathcal{G}_\psi$ with $\mathcal{G} = \{\neg\psi\}$

Above, $\texttt{Grounding}_{\mathcal{K}}(r)$ is the set containing the grounded rule $\sigma(\texttt{body}(r)) \to \sigma(\texttt{head}(r))$ for every trigger $\tau = \langle r, \sigma \rangle$ over $\texttt{Chase}(\mathcal{K})$. Given a set of groups $\mathfrak{F}$, we denote by $\texttt{KBs}(\mathfrak{F})$ the knowledge base built by using all facts and rules that are identifiers of the groups in $\mathfrak{F}$. Proposition 1 establishes a precise correspondence between the explanations for $\mathcal{K} \models \psi$ and the group-MUSes of $\texttt{GSAT}(\mathcal{K} \wedge \neg\psi)$.

Proposition 1. $\texttt{Expl}(\mathcal{K}, \psi) = \bigcup_{\mathfrak{F} \in \texttt{GMUS}(\texttt{GSAT}(\mathcal{K} \wedge \neg\psi))} \texttt{KBs}(\mathfrak{F})$.

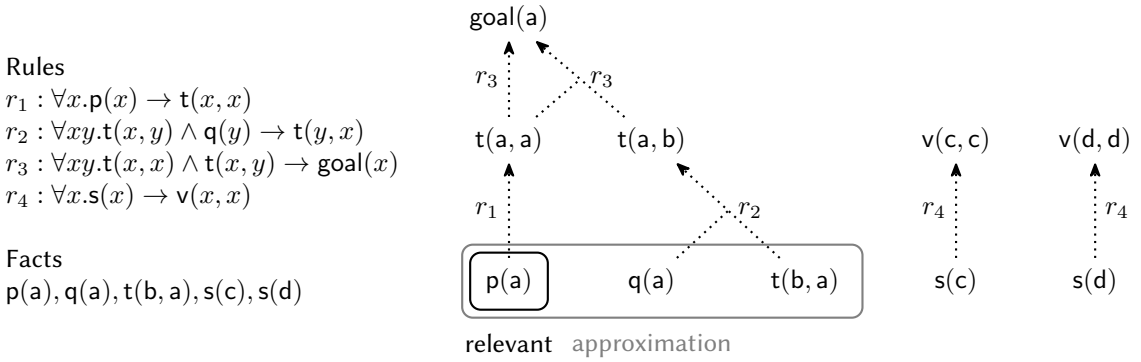

Figure 3: Relevance: Exact vs Approximate

3. Knowledge-Base Filtering via Relevance

To explain an entailment of a knowledge base, it is often the case that only a portion of the knowledge base's facts and rules are necessary or *relevant* to that entailment. In this section, we present a formal definition of relevance and introduce a rule-based technique for filtering out facts that are irrelevant for explaining a given fact. This technique can drastically reduce the size of the encoded formula to be solved.

Relevance We say that a fact ψ is *relevant* for the entailment of a fact φ in a knowledge base $\mathcal{K}$ if there exists an explanation $\mathcal{K}' \in \texttt{Expl}(\mathcal{K}, \varphi)$ that contains ψ. This definition naturally extends to rules. Consider the knowledge base in Figure 3 with facts $\mathcal{F} = \{\mathsf{p}(\mathsf{a}), \mathsf{q}(\mathsf{a}), \mathsf{t}(\mathsf{b},\mathsf{a}), \mathsf{s}(\mathsf{c}), \mathsf{s}(\mathsf{d})\}$ and rules $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$. The entailment $\varphi = \mathsf{goal}(\mathsf{a})$ can be derived in two ways. The first is by applying r_1, which yields $\mathsf{t}(\mathsf{a},\mathsf{a})$; this atom allows for the application of r_3 to derive φ. The second is by applying r_3 on the results of r_1 and r_2. The fact φ has, however, *only one explanation*, namely, $\mathcal{K}' = \langle\{r_1, r_3\}, \{\mathsf{p}(\mathsf{a})\}\rangle$. Indeed, $\mathcal{K}'' = \langle\{r_1, r_2, r_3\}, \{\mathsf{p}(\mathsf{a}), \mathsf{q}(\mathsf{a}), \mathsf{t}(\mathsf{b},\mathsf{a})\}\rangle$ is *not* minimal as $\mathcal{K}' \subset \mathcal{K}''$ (see Definition 1). As a result, with $\texttt{Expl}(\mathcal{K}, \varphi) = \{\mathcal{K}'\}$, the only elements relevant to the entailment of φ in $\mathcal{K}$ are $\mathsf{p}(\mathsf{a})$ and r_1, r_3. Despite its apparent simplicity, it turns out that deciding relevance is hard. This holds true even if one focuses only on deciding the relevance of facts, for a fixed rule set. By a reduction from SAT, we have the following proposition. Proofs can be found in Appendix A.

Proposition 2. *For a fixed ruleset $\mathcal{R}$, deciding if a fact $\psi \in \mathcal{F}$ is relevant for φ on $\langle\mathcal{R}, \mathcal{F}\rangle$ is NP-complete.*

Approximating Relevance Considering this hardness result, we turn our attention to the task of *approximating* relevant facts and rules. This is achieved using a two-step rule-based approach: a static step which preprocesses the input knowledge base, followed by a dynamic step for tracing each individual entailment. The static step builds an *entailment graph* which tracks all relationships between any entailed atoms and rules. This entailment graph allows us to compute an approximation of relevant facts. For instance, as illustrated in Figure 3, the relevance approximation includes atoms $\mathsf{p}(\mathsf{a}), \mathsf{q}(\mathsf{a}), \mathsf{t}(\mathsf{b},\mathsf{a})$ and rules r_1, r_2, r_3.

We now present our approach and give a construction for the entailement graph based on *rules with function symbols*, which can be deployed on reasoners supporting the formalism. Before detailing each step, we introduce the following notation. Given a predicate P, we denote by P^+ and f_P a fresh predicate and a function unique for P, respectively; below, the same will hold for predicates B and H. Moreover, given a rule r we respectively denote by E_r and f_r a fresh predicate and a function unique for r.

Static step (entailment graph building) Given a knowledge base $\mathcal{K} = \langle\mathcal{R}, \mathcal{F}\rangle$, we compute $\mathcal{S} = \texttt{Chase}(\langle\texttt{EntGraph}(\mathcal{R}), \mathcal{F}\rangle)$ where $\texttt{EntGraph}(\mathcal{R})$ is the minimal set containing the following rules:

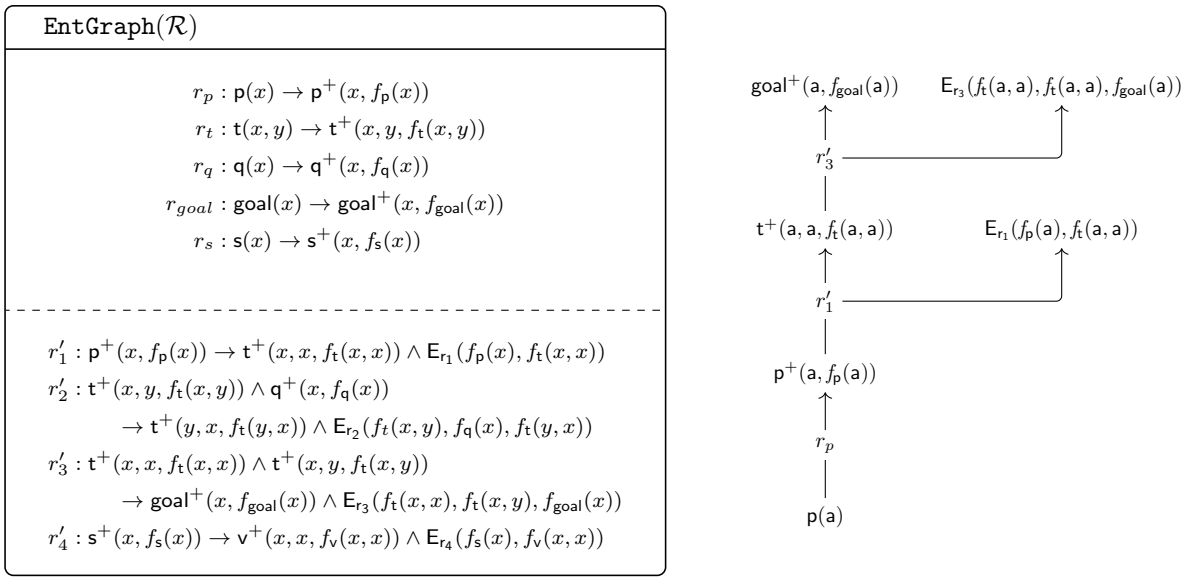

Figure 4: Static Step Rules and Inferences

1. $P(\bar{x}) \to P^+(\bar{x}, f_P(\bar{x}))$ for every predicate P occurring in $\mathcal{F}$
2. $\bigwedge_{i=1}^n B_i^+(\bar{x}_i, y_i) \to H^+(\bar{x}, f_H(\bar{x})) \wedge \mathsf{E}_r(y_1, \ldots, y_n, f_H(\bar{x}))$
   for every rule $r = \bigwedge_{i=1}^n B_i(\bar{x}_i) \to H(\bar{x}) \in \mathcal{R}$

The aim of the static step is to produce the entailment graph, i.e. a hypergraph representing the links (in terms of entailment) between the atoms. In order to do that, we first need to be able to reference the atoms in $\texttt{Chase}(\mathcal{K})$. This is done by rules of type (1) which infer, for any atom $P(\bar{t})$ of arity k, an atom $P^+(\bar{t}, f_P(\bar{t}))$ of arity $k+1$ where $f_P(\bar{t})$ is a functional term representing $P(\bar{t})$ itself. Then, every rule r is transformed into a rule of type (2) which, once applied, uses the functional terms to link all images of the body atoms to the image of the head atom via an hyperedge relation E_r proper to r. Let us emphasise that this step is done *only once*, for every input $\mathcal{K}$.

Figure 4 illustrates the rules for the static step $\texttt{EntGraph}(\mathcal{R})$ for the rule base of Figure 3, as well as their applications starting from $\mathsf{p}(\mathsf{a})$. First, the rule r_p produces the atom $\mathsf{p}^+(\mathsf{a}, f_\mathsf{p}(\mathsf{a}))$, where $f_\mathsf{p}(\mathsf{a})$ represents $\mathsf{p}(\mathsf{a})$. Consider then rule r_1. This is transformed in the following way: the body $\mathsf{p}(x)$ and the head $\mathsf{t}(x,x)$ are respectively replaced by $\mathsf{p}^+(x, f_\mathsf{p}(x))$ and $\mathsf{t}^+(x,x, f_\mathsf{t}(x,x))$, and an hyperedge atom $\mathsf{E}_{\mathsf{r}_1}(f_\mathsf{p}(x), f_\mathsf{t}(x,x))$ is added to the head to represent the link between the two atoms in the entailment graph. Note that this edge atom uses the functional terms corresponding to the atoms appearing in the rule. Since we have $\mathsf{p}^+(\mathsf{a}, f_\mathsf{p}(\mathsf{a}))$, the rule r'_1 produces both $\mathsf{t}^+(\mathsf{a},\mathsf{a}, f_\mathsf{t}(\mathsf{a},\mathsf{a}))$ and the edge predicate $\mathsf{E}_{\mathsf{r}_1}(f_\mathsf{p}(\mathsf{a}), f_\mathsf{t}(\mathsf{a},\mathsf{a}))$ representing the fact that $\mathsf{p}^+(\mathsf{a}, f_\mathsf{p}(\mathsf{a}))$ is used to derive $\mathsf{t}^+(\mathsf{a},\mathsf{a}, f_\mathsf{t}(\mathsf{a},\mathsf{a}))$. Finally, r'_3 is obtained from r_3 in a similar way, and it is used to derive $\mathsf{goal}^+(\mathsf{a}, f_\mathsf{goal}(\mathsf{a},\mathsf{a}))$.

Dynamic Step (tracing) For any fact φ to explain, we define $\mathcal{D} = \texttt{Chase}(\langle \texttt{relPropag}(\mathcal{R}, \varphi), \mathcal{S} \rangle)$ where $\mathcal{S}$ is the result of the static step and $\texttt{relPropag}(\mathcal{R}, \varphi)$ is the minimal set containing the following rules:

1. $\to \mathsf{Rel}(f_P(\bar{x}))$ if $\varphi = P(\bar{x})$
2. $\mathsf{E}_\mathsf{r}(y_1, \ldots, y_n, z) \wedge \mathsf{Rel}(z) \to \mathsf{Rel}(y_1) \wedge \ldots \wedge \mathsf{Rel}(y_n) \wedge \mathsf{RelEdge}(f_r(y_1, \ldots, y_n, z))$
   for every rule r in $\mathcal{R}$

Note first that rule (1) has an empty body; this is used to bootstrap the tracing for φ. Furthermore, for rules of type (2), we consider that E_r and f_r are the fresh predicate and function unique for r introduced

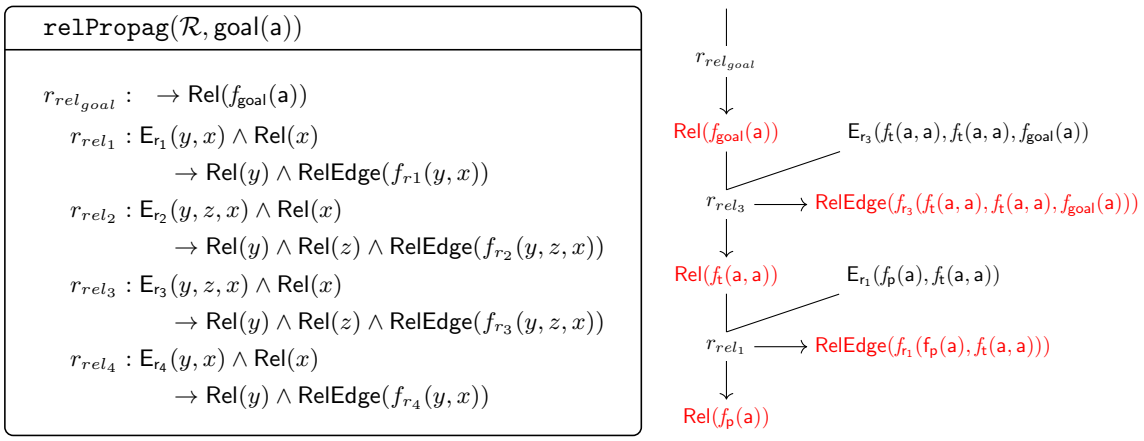

Figure 5: Dynamic Step Rules and Inferences

in the static step, and Rel and RelEdge are reserved predicates. Intuitively, these rules recursively trace a fact back to the ancestor atoms which belong to the input fact base.

Figure 5 illustrates the rules for the dynamic step $\texttt{relPropag}(\mathcal{R}, \varphi)$ for the rule base of Figure 3 and the entailment $\varphi = \mathsf{goal}(\mathsf{a})$, as well as their applications. In a nutshell, the atom to be explained is marked as relevant thanks to the (empty body) rule $\rightarrow \mathsf{Rel}(f_{\mathsf{goal}}(\mathsf{a}))$. Then the relevance relation is propagated back using the edge atoms produced during the static step. In order to do this, we need the "propagation rule" r_{rel_1}-r_{rel_4}. For instance, since we have $\mathsf{E}_{\mathsf{r}_3}(f_{\mathsf{t}}(\mathsf{a},\mathsf{a}), f_{\mathsf{t}}(\mathsf{a},\mathsf{a}), f_{\mathsf{goal}}(\mathsf{a}))$ from the static step, and $\mathsf{goal}(\mathsf{a})$ is relevant (i.e., $\mathsf{Rel}(f_{\mathsf{goal}}(\mathsf{a}))$), then by rule r_{rel_3} we have that both $\mathsf{t}(\mathsf{a},\mathsf{a})$ and r_3 are relevant (i.e, $\mathsf{Rel}(f_{\mathsf{t}}(\mathsf{a},\mathsf{a}))$ and $\mathsf{RelEdge}(f_{\mathsf{r}_3}(f_{\mathsf{t}}(\mathsf{a},\mathsf{a}), f_{\mathsf{t}}(\mathsf{a},\mathsf{a}), f_{\mathsf{goal}}(\mathsf{a})))$, respectively). The same process applies for deeming $\mathsf{p}(\mathsf{a})$ and rule r_1 as relevant by the application of r_{rel_1}.

Finally, for $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$, we define the approximation of the relevant KB as $\texttt{SupRel}_{\varphi}(\mathcal{K}) = \langle \mathcal{R}', \mathcal{F}' \rangle$ where $\mathcal{F}' = \{P(\bar{t}) \in \mathcal{F} \mid \mathcal{D} \models \mathsf{Rel}(f_P(\bar{t}))\}$ and $\mathcal{R}' = \{r \in \mathcal{R} \mid \mathcal{D} \models \mathsf{RelEdge}(f_r(t_1, \ldots, t_n, t'))\}$. Note that both $\mathcal{S}$ and $\mathcal{D}$ knowledge bases always admit a finite (terminating) chase. Soundness and completeness below state that this approximation provides the same explanations as the exact relevance.

Proposition 3. $\texttt{Expl}(\mathcal{K}, \varphi) = \texttt{Expl}(\texttt{SupRel}_{\varphi}(\mathcal{K}), \varphi)$

4. Experimental Analysis

We have implemented our approach by using the MARCO tool for group-MUS enumeration [11] (Section 2) and InteGraal [14] for approximating relevance (Section 3).

Benchmarks We selected 24 interesting ontologies from the MOWL corpus [15]. These ontologies were in the EL profile and have been translated into Datalog programs using the Java OWL API [16] and the translation presented in [17]. We made sure that each ontology has at least 5 extensional facts and produces at least 5 intensional facts. For each ontology, we picked 5 entailment queries to explain among the atoms with the greatest reasoning depth (and that are hence, intuitively, more challenging to explain), that is, atoms that are derived in the last (breadth-first) step of the saturation of the ontology.

Protocol For each ontology, we compute the entailment graph using InteGraal [14] and the rules for the *static step* presented in Section 3; recall that this step is done only once per ontology. Subsequently, for each entailment query, we again use InteGraal [14] and the rules for the *dynamic step* presented in Section 3 to compute a knowledge base that consists only of the part of the input ontology relevant to that particular entailment query. This relevant knowledge base is then translated into a GSAT formula as described in Section 2 and fed to the MARCO tool [11]. The tool returns all MUSes which, when

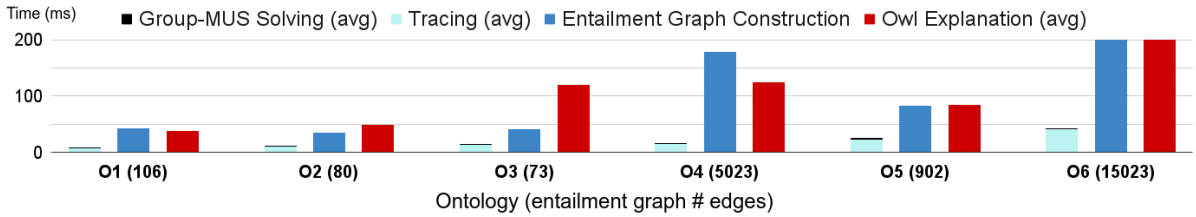

Figure 6: Query Explanation Performances

translated back to Datalog knowledge base statements, constitute the explanations. We compared against the OWL API [16] which is the only tool we are aware of which computes kb-support explanations (i.e., ontology axioms corresponding to facts and rules). We implemented our benchmarking protocols using the B-Runner library [18] and made them available online [19].

Setup All our experiments are run on a server with AMD Ryzen 9 3900XT 12-Core Processor @4.7GHz, 128GB of RAM @2.4GHz, and 2TB of SSD NVME.

Analysis Figure 6 reports the time taken by the *static* step, which corresponds to the entailment graph construction, vs the *dynamic* step which is further divided into filtering (via InteGraal) and MUS enumeration (via MARCO). For the dynamic step, we report the average time for the 5 queries we considered. Times are in milliseconds.

As expected, we can observe that the time taken by the *static step* (even if within tenths of a second) is significantly longer than that of the dynamic step. Interestingly, we found that the time required for constructing the entailment graph depends on the number of possible rule groundings as well as on the *depth* (in a breadth-first sense) of the chase. This cost, being fixed, is however amortised when multiple explanation queries are submitted.

Concerning the *dynamic step*, we can observe that this is much faster than the static step, more in the order of tens of milliseconds. Again, we found that a crucial aspect here is the depth of the derived atom: the deeper an atom is in the entailment graph the more one has to propagate relevance. Our results also indicate that the group-MUS enumerator is very fast (in the order of milliseconds), which further validates this approach for computing explanations. Note also that the generality of our method makes that it can be implemented with any reasoner supporting the rules described in Section 3, and any group-MUS enumerator.

We conclude with a comparison with the explanation facility provided by the OWL API. We can see that our approach is generally more competitive if we consider only the dynamic step, and can even be more competitive depending on the cases for one-of explanations including both the static and dynamic step as a single operation.

5. Related Work & Conclusion

We have introduced the first comprehensive approach for computing kb-support explanations for fact entailments over Datalog knowledge bases. Our method leverages on group-MUS enumeration for computing explanations as well as on a rule-based approach to filter irrelevant atoms for the task. Our approach has been implemented and experimentally evaluated.

Our work considers explanations in the form of kb-support (a.k.a, justifications with ABoxes), which, to the best of our knowledge, has not been investigated so far for Datalog [9, 8, 10]. Indeed, existing methods for Datalog focus on the notion of *why-provenance* which corresponds to that of fact-support. The recent work of [8] studies the use of SAT-solvers for computing why-provenance on Datalog based on so called *non-ambiguous* proof trees. Rule-based approaches for on-demand computation of Datalog provenance have been studied in [10]. The recent work of [5] considers provenance computation for

the EL Description Logic [20, 21]. In the context of answer set programming (ASP), the work of [21] considers a notion of graph-explanation; this can be seen as an extension of fact-support including all inferred facts (but no rules). While these approaches are close to ours in spirit, the technical development remains different. Notably, considering kb-supports allows us to obtain more explanations than why-provenance, and thus a better understanding of the entailed facts. On the other hand, this also required us to establish a new correspondence between group-MUSes and kb-support explanations.

In the context of Description Logics, reasoning explanations have long been studied. Justifications, also called *minimal axiom sets*, and their computation are referred to as *axiom pinpointing*. The approaches closest to ours are the following. [1] and [2] consider the problem of computing axiom explanation for EL TBoxes (hence, without data). [1] studies the complexity of computing relevant axioms while [2] presents a reduction to GMUS enumeration. However, in contrast to our work, none of these approaches consider data (ABoxes). The problem of filtering the knowledge base to a set of relevant facts has been considered in [8, 10, 3, 4]. All of these approaches employ a notion akin to that of a dependency graph. However, our work differs in that it considers a different notion of explanation. Moreover, our work is the only one that studies the complexity of deciding the relevance of atoms (NP-complete).

Through this work we demonstrated the practical applicability of group-SAT solvers for knowledge bases, paving the way for more efficient and explainable AI systems. Future work includes the extension of our setting to richer rule languages, including suitable forms of functions, negation, and disjunction.

Acknowledgments

This work was supported by the Inria-DFKI bilateral project R4Agri and by the ANR project CQFD (ANR-18-CE23-0003).

References

[1] F. Baader, R. Penaloza, B. Suntisrivaraporn, Pinpointing in the description logic, in: Annual Conference on Artificial Intelligence, Springer, 2007, pp. 52–67.

[2] N. Manthey, R. Peñaloza, S. Rudolph, Satpin: Axiom pinpointing for lightweight description logics through incremental sat, KI-Künstliche Intelligenz 34 (2020) 389–394.

[3] Z. Zhou, G. Qi, B. Suntisrivaraporn, A new method of finding all justifications in owl 2 el, in: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), volume 1, IEEE, 2013, pp. 213–220.

[4] Z. Zhou, G. Qi, A dependency-graph based approach for finding justification in owl 2 el, Intelligent Data Analysis 22 (2018) 1315–1335.

[5] S. Borgwardt, S. Breuer, A. Kovtunova, Computing abox justifications for query answers via datalog rewriting., in: Description Logics, 2023.

[6] S. Abiteboul, R. Hull, V. Vianu, Foundations of databases, volume 8, Addison-Wesley Reading, 1995.

[7] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, 2018, pp. 5511–5519. URL: https://doi.org/10.24963/ijcai.2018/777. doi:10.24963/IJCAI.2018/777.

[8] M. Calautti, E. Livshits, A. Pieris, M. Schneider, Computing the why-provenance for datalog queries via SAT solvers, in: M. J. Wooldridge, J. G. Dy, S. Natarajan (Eds.), Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, AAAI Press, 2024, pp. 10459–10466. URL: https://doi.org/10.1609/aaai.v38i9.28914. doi:10.1609/AAAI.V38I9.28914.

[9] C. Bourgaux, P. Bourhis, L. Peterfreund, M. Thomazo, Revisiting semiring provenance for datalog, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 - August 5, 2022, 2022. URL: https://proceedings.kr.org/2022/10/.

[10] A. Elhalawati, M. Krötzsch, S. Mennicke, An existential rule framework for computing why-provenance on-demand for datalog, in: G. Governatori, A. Turhan (Eds.), Rules and Reasoning - 6th International Joint Conference on Rules and Reasoning, RuleML+RR 2022, Berlin, Germany, September 26-28, 2022, Proceedings, volume 13752 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 146–163. URL: https://doi.org/10.1007/978-3-031-21541-4_10. doi:10.1007/978-3-031-21541-4_10.

[11] M. H. Liffiton, A. Previti, A. Malik, J. Marques-Silva, Fast, flexible MUS enumeration, Constraints An Int. J. 21 (2016) 223–250. URL: https://doi.org/10.1007/s10601-015-9183-0. doi:10.1007/S10601-015-9183-0.

[12] C. Beeri, M. Y. Vardi, A proof procedure for data dependencies, J. ACM 31 (1984) 718–741. URL: https://doi.org/10.1145/1634.1636. doi:10.1145/1634.1636.

[13] M. H. Liffiton, K. A. Sakallah, Algorithms for computing minimal unsatisfiable subsets of constraints, J. Autom. Reason. 40 (2008) 1–33. URL: https://doi.org/10.1007/s10817-007-9084-z. doi:10.1007/S10817-007-9084-Z.

[14] J.-F. Baget, P. Bisquert, M. Leclère, M.-L. Mugnier, G. Pérution-Kihli, F. Tornil, F. Ulliana, InteGraal: a Tool for Data-Integration and Reasoning on Heterogeneous and Federated Sources, in: BDA 2023 - 39e Conférence sur la Gestion de Données – Principes, Technologies et Applications, Montpellier, France, 2023. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-04304601.

[15] N. Matentzoglu, B. Parsia, The Manchester OWL Corpus (MOWLCorp), original serialisation, 2014. URL: https://doi.org/10.5281/zenodo.10851. doi:10.5281/zenodo.10851.

[16] M. Horridge, S. Bechhofer, The OWL API: A java API for OWL ontologies, Semantic Web 2 (2011) 11–21. URL: https://doi.org/10.3233/SW-2011-0025. doi:10.3233/SW-2011-0025.

[17] D. Carral, J. Zalewski, P. Hitzler, An efficient algorithm for reasoning over OWL EL ontologies with nominal schemas, J. Log. Comput. 33 (2023) 136–162. URL: https://doi.org/10.1093/logcom/exac032. doi:10.1093/LOGCOM/EXAC032.

[18] F. Ulliana, P. Bisquert, A. Charoensit, R. Colin, F. Tornil, Q. Yeche, Collaborative benchmarking with b-runner, in: Rules and Reasoning - 8th International Joint Conference on Rules and Reasoning, RuleML+RR 2024, Bucarest, Romania, September 16-18, 2024, Proceedings, Lecture Notes in Computer Science, 2024.

[19] Artifacts for Rule-aware Datalog Fact Explanation Using Group-SAT Solver, https://gitlab.inria.fr/boreal-artifacts/ruleml2024, 2024.

[20] T. Eiter, T. Geibinger, Explaining answer-set programs with abstract constraint atoms., in: IJCAI, 2023, pp. 3193–3202.

[21] M. Alviano, L. L. Trieu, T. C. Son, M. Balduccini, Explanations for answer set programming, arXiv preprint arXiv:2308.15879 (2023).

A. Proofs for Section 3

FACT RELEVANCE($\mathcal{R}$)

INSTANCE: A fact set $\mathcal{F}$, a fact φ and a fact $\psi \in \mathcal{F}$.
QUESTION: Is ψ *relevant* for the entailment of φ with respect to $\langle \mathcal{R}, \mathcal{F} \rangle$?

Lemma 1. *FACT RELEVANCE($\mathcal{R}$) is in NP.*

Proof. We consider an oracle that, for a fixed $\mathcal{R}$, can decide in polynomial time FACT ENTAILMENT($\mathcal{R}$), that is, whether $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi$ for a given fact set $\mathcal{F}$ and fact φ. We then consider a non-deterministic Turing Machine that *accepts* on input $\langle \mathcal{F}, \varphi, \psi \rangle$ if and only if ψ is relevant for φ with $\langle \mathcal{R}, \mathcal{F} \rangle$ and does the following.

1. Non-deterministically guesses the sets $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{R}' \subseteq \mathcal{R}$ such that $\psi \in \mathcal{F}'$.
2. Checks with the oracle on FACT ENTAILMENT($\mathcal{R}$) whether $\langle \mathcal{R}', \mathcal{F}' \rangle \models \varphi$. If it does not, *rejects*.
3. Checks the minimality of $\langle \mathcal{R}', \mathcal{F}' \rangle$. If there exists $\langle \mathcal{R}'', \mathcal{F}'' \rangle \subseteq \langle \mathcal{R}', \mathcal{F}' \rangle$ with $|\mathcal{F}''| = |\mathcal{F}'| - 1$ or $|\mathcal{R}''| = |\mathcal{R}'| - 1$ such that $\langle \mathcal{R}'', \mathcal{F}'' \rangle \models \varphi$ (checked with the oracle) then *rejects*, otherwise *accepts*.

(Soundness) Given a fact set $\mathcal{F}$, a fact φ and some fact $\psi \in \mathcal{F}$, we show that if the TM *accepts* on input $\langle \mathcal{K}, \varphi, \psi \rangle$ then ψ is relevant for φ with $\langle \mathcal{R}, \mathcal{F} \rangle$. By hypothesis the TM *accepts* so there is a pair $\langle \mathcal{R}', \mathcal{F}' \rangle \subseteq \langle \mathcal{R}, \mathcal{F} \rangle$ such that $\psi \in \mathcal{F}'$, $\langle \mathcal{R}', \mathcal{F}' \rangle \models \varphi$ and $\langle \mathcal{R}'', \mathcal{F}'' \rangle \not\models \varphi$ for every $\mathcal{F}'' \subset \mathcal{F}'$ and $\mathcal{R}'' \subset \mathcal{R}'$. Hence, by the definitions of explanations and relevance, $\langle \mathcal{R}', \mathcal{F}' \rangle$ is an explanation for φ containing ψ, thus ψ is relevant for φ.

(Completeness) Given a fact set $\mathcal{F}$, a fact φ and some fact $\psi \in \mathcal{F}$, we show that if ψ is relevant for φ then the TM *accepts* on input $\langle \mathcal{K}, \varphi, \psi \rangle$. By hypothesis ψ is relevant for φ so there is an explanation $\mathcal{E} = \langle \mathcal{R}', \mathcal{F}' \rangle$ for φ containing ψ. Hence, $\psi \in \mathcal{F}'$ and $\langle \mathcal{R}', \mathcal{F}' \rangle \models \varphi$. At step 1 the TM guesses the explanation $\mathcal{E} = \langle \mathcal{R}', \mathcal{F}' \rangle$ and by construction of $\mathcal{E}$ the oracle returns "yes" at step 2. Again, by hypothesis $\langle \mathcal{R}', \mathcal{F}' \rangle$ is minimal thus the oracle answers "no" for every $\langle \mathcal{R}'', \mathcal{F}'' \rangle \subset \langle \mathcal{R}', \mathcal{F}' \rangle$ of size $|\langle \mathcal{R}', \mathcal{F}' \rangle| - 1$ checked in step 3, and the TM *accepts*.

(Termination) As step one is a non-deterministic guess of subsets of $\mathcal{F}$ and $\mathcal{R}$ and then we use the P oracle a linear number of times in the size of the guessed subsets, FACT RELEVANCE($\mathcal{R}$) is in $NP^P = NP$. □

Lemma 2. *FACT RELEVANCE($\mathcal{R}$) is NP-hard.*

Proof. We show that FACT RELEVANCE($\mathcal{R}$) is NP-hard via a reduction from SAT.

(Translation) For a CNF formula $\Phi = C_1 \wedge \ldots \wedge C_m$ defined using the variables $\{V_1, \ldots, V_n\}$, we produce a knowledge base $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ with nullary predicates Source and Target such that Φ is satisfiable if and only if Source is relevant for the entailment of Target. The knowledge base $\mathcal{K}$ is built as follows:

$$\mathcal{F} = \{\mathsf{T}(v_i), \mathsf{F}(v_i) \mid 1 \leq i \leq n\} \cup \{\mathsf{Source}\}$$

$$\mathcal{R} = \{\mathsf{T}(v_k) \to \mathsf{T}(c_i) \mid 1 \leq i \leq m \text{ and every positive literal } V_k \text{ in } C_i\} \cup \tag{1}$$

$$\{\mathsf{F}(v_k) \to \mathsf{T}(c_i) \mid 1 \leq i \leq m \text{ and every negative literal } \neg V_k \text{ in } C_i\} \cup \tag{2}$$

$$\{\mathsf{Source} \wedge \bigwedge\nolimits_{i=1}^{m} \mathsf{T}(c_i) \to \mathsf{Target}\} \cup \tag{3}$$

$$\{\mathsf{T}(x) \wedge \mathsf{F}(x) \to \mathsf{Target}\} \tag{4}$$

Note that in the construction above, we introduced a unique constant v_i for each variable V_i and a unique constant c_i for each clause C_i in Φ. Note that therefore the only rule with variables is (4).

(Soundness) We show that if Source is relevant for the entailment of Target with $\mathcal{K}$ then the CNF formula Φ is satisfiable.

ⓐ By hypothesis there is an explanation $\langle \mathcal{R}', \mathcal{F}' \rangle \subseteq \langle \mathcal{R}, \mathcal{F} \rangle$ such that $\mathsf{Source} \in \mathcal{F}'$ and $\langle \mathcal{R}', \mathcal{F}' \rangle \models \mathsf{Target}$.

ⓑ We show first that for every constant v_i appearing in $\mathcal{F}'$, we have that $\mathsf{T}(v_i) \in \mathcal{F}'$ if and only if $\mathsf{F}(v_i) \notin \mathcal{F}'$. Indeed, suppose $\mathsf{T}(v_i), \mathsf{F}(v_i) \in \mathcal{F}'$ for some constant v_i. Then, with rule (4) we have $\langle \mathcal{R}', \{\mathsf{T}(v_i), \mathsf{F}(v_i)\} \rangle \models \mathsf{Target}$. This contradicts the fact that $\mathcal{F}'$ is a minimal set that, together with $\mathcal{R}'$, entails Target.

ⓒ Let $\sigma_{\mathcal{F}'}$ be the assignment such that $\sigma_{\mathcal{F}'}(V_i) = \textit{True}$ if and only if $\mathsf{T}(v_i) \in \mathcal{F}'$ for every variable V_i in Φ and its corresponding constant v_i in $\mathcal{K}$.

ⓓ From ⓑ, we know that (4) cannot be applied in $\mathcal{F}'$. Therefore, the rule (3) has been applied to infer Target.

Hence, $\langle \mathcal{R}', \mathcal{F}' \rangle \models \mathsf{T}(c_i)$ for every $1 \leq i \leq m$.

ⓔ For $1 \leq i \leq m$ we know that $\mathsf{T}(c_i)$ is either inferred by a rule from (1) or from (2).

If a rule of the form (1) has been applied then there is a constant v_k such that $\mathsf{T}(v_k) \in \mathcal{F}'$ and $\mathsf{T}(v_k) \rightarrow \mathsf{T}(c_i) \in \mathcal{R}'$. Let V_k be the corresponding variable of v_k and C_i the clause corresponding to c_i in Φ, then by the construction of the rules in (1) we have that $V_k \in C_i$ and by ⓒ we know that $\sigma_{\mathcal{F}'}(V_k) = \textit{True}$, thus $\sigma_{\mathcal{F}'}(C_i) = \textit{True}$.

If a rule of the form (2) has been applied then there is a constant v_k such that $\mathsf{F}(v_k) \in \mathcal{F}'$ and $\mathsf{F}(v_k) \rightarrow \mathsf{T}(c_i) \in \mathcal{R}'$. Let V_k be the corresponding variable of v_k and C_i the clause corresponding to c_i in Φ, then by the construction of the rules in (2) we have that $\neg V_k \in C_i$ and by ⓒ we know that $\sigma_{\mathcal{F}'}(V_k) = \textit{False}$ thus $\sigma_{\mathcal{F}'}(C_i) = \textit{True}$.

Hence, $\sigma_{\mathcal{F}'}(C_i) = \textit{True}$ for every clause $C_i \in \Phi$, thus $\sigma(\Phi) = \textit{True}$ and Φ is satisfiable.

(Completeness) We show that if Φ is satisfiable then Source is relevant for Target with $\mathcal{K}$.

ⓐ By hypothesis, there is an assignment σ such that σ is a model of Φ. Consider the set of facts $\mathcal{F}' = \{\mathsf{Source}\} \cup \{\mathsf{T}(v_i) \mid \sigma(V_i) = \textit{True}\} \cup \{\mathsf{F}(v_i) \mid \sigma(V_i) = \textit{False}\} \subseteq \mathcal{F}$.

ⓑ By hypothesis, we have $\sigma(C_i) = True$ for every clause C_i in Φ and two cases hold.

There is a positive literal $V_k \in C_i$ such that $\sigma(V_k) = True$. In this case, let v_k and c_i be the constants corresponding to V_k and C_i respectively. By ⓐ we have that $\mathsf{T}(v_k) \in \mathcal{F}'$, and $\mathsf{T}(v_k) \rightarrow \mathsf{T}(c_i) \in \mathcal{R}$ by the construction of rules in (1).

There is a negative literal $\neg V_k \in C_i$ such that $\sigma(V_k) = False$. Let v_k and c_i be the constants corresponding to V_k and C_i respectively. By ⓐ we have that $\mathsf{F}(v_k) \in \mathcal{F}'$, and $\mathsf{F}(v_k) \rightarrow \mathsf{T}(c_i) \in \mathcal{R}$ by the construction of rules in (2).

In both cases, $\langle \mathcal{R}, \mathcal{F}' \rangle \models \mathsf{T}(c_i)$ for every $1 \leq i \leq m$.

ⓒ Let $\mathcal{F}''$ be a minimal subset of $\mathcal{F}'$ and $\mathcal{R}'$ a minimal subset of $\mathcal{R}$ (obtained in linear time by *deletion*, i.e., by removing on fact/rule until the entailment is lost) such that $\langle \mathcal{R}', \mathcal{F}'' \rangle \models \mathsf{Target}$. We know that (3) and (4) are the only rules that can infer Target. However, (4) can not be applied on $\mathcal{F}''$. Indeed, $\mathcal{F}'$ is built from an assignment σ and therefore either $\mathsf{T}(v_i)$ or $\mathsf{F}(v_i)$ belongs to $\mathcal{F}''$ for any constant v_i corresponding to a variable V_i in Φ. Hence the rule (3) has been applied to derive Target.

ⓓ To conclude, note that to apply (3) the atom Source must belong to $\mathcal{F}''$ and $\langle \mathcal{R}', \mathcal{F}' \rangle$ is an explanation of Target. Hence, Source is relevant for Target.

(Polynomiality) We show the size of $\mathcal{K}$ is polynomial with respect to the size of Φ. In the fact set $\mathcal{F}$ we have $2n + 1$ atoms with n being the number of propositional variables in Φ. Then the rule set contains at most $k \times m + 2$ rules where m is the number of clauses in Φ and k the largest number of literals in a clause. Thus the instance $\mathcal{K}$ is polynomial with respect to Φ and the translation can be computed in polynomial time. □