Explaining Critical Situations Over Sensor Data Streams Using Proofs and Natural Language

Stefan Borgwardt[1], Vera Demberg[2,3], Mayank Jobanputra[2], Alisa Kovtunova[1] and Duy Nhu[1]

[1] *Institute of Theoretical Computer Science, TU Dresden, Germany*
[2] *Department of Computer Science, Saarland University, Germany*
[3] *Department of Language Science and Technology, Saarland University, Germany*

Abstract

This paper describes ongoing interdisciplinary work on integrating logical reasoning and natural language generation, in order to support a drone operator in the task of supervising autonomous drones, e.g. for package delivery. In previous work, justifications were computed to explain description logic (DL) reasoning, and these justifications were then used as the basis for generating a verbal situation description. Here, we consider full proofs instead of justifications, as well as temporal and concrete domains, in order to generate more informative natural language explanations. Following a method for extracting DL proofs, we (i) implemented inference tracing for the DatalogMTL reasoner MeTeoR, and (ii) used the Evee library (a tool collection for explaining DL entailments) to extract small proofs for entailed facts. We provide an empirical evaluation of the tracing and proof extraction phases, which reveals only a minor increase in the computational costs compared to the original MeTeoR reasoner. Finally, we discuss our preliminary findings on using these proofs as input to a natural language generation module that explains the reasons behind entailed facts to a user.

Keywords

Proofs, DatalogMTL, Concrete domain reasoning, Temporal reasoning, Verbalization

1. Introduction

Drone technology and drone control have recently advanced rapidly, to the point that consumer drones with advanced sensors and improved control algorithms are commonplace [1], e.g. in aerial surveys, mapping, aerial movies or rescue missions. As drones are used for an increasing range of tasks, it becomes more important to interact with them. To enable these interactions, it is essential to devise a setup that can flexibly process a variety of data collected by the drone during its flight, reason over it, and convey the important information to the user at *runtime*, as well as provide a full report at *inspection time*. Here, we focus on extracting and processing relevant information from sensor data records in order to perform a controlled handover from an autonomous drone to a human drone pilot (see Figure 1). In this setup, the content selection task is hard since only critical information, not all available information, should be mentioned at handover time. Additionally, it is desirable that the system generalizes well to diverse as well as unseen environments during its operation.

To this end, we propose that this internal reasoning is made using FOL theories (such as description logic ontologies [2] and logic programs [3]) and their temporal extensions. This provides us with many benefits. First, expert-designed logic theories allow for more flexibility

DL 2024: 37th International Workshop on Description Logics, June 18–21, 2024, Bergen, Norway
stefan.borgwardt@tu-dresden.de (S. Borgwardt); vera@coli.uni-saarland.de (V. Demberg); mayank@lst.uni-saarland.de (M. Jobanputra); alisa.kovtunova@tu-dresden.de (A. Kovtunova); duy.nhu@mailbox.tu-dresden.de (D. Nhu)
https://lat.inf.tu-dresden.de/~stefborg/ (S. Borgwardt); https://www.uni-saarland.de/lehrstuhl/demberg/members/verademberg.html (V. Demberg); https://www.uni-saarland.de/lehrstuhl/demberg/members/mayank-jobanputra.html (M. Jobanputra); https://lat.inf.tu-dresden.de/~alisa/ (A. Kovtunova)
0000-0003-0924-8478 (S. Borgwardt); 0000-0002-8834-0020 (V. Demberg); 0000-0001-9936-0943 (A. Kovtunova)

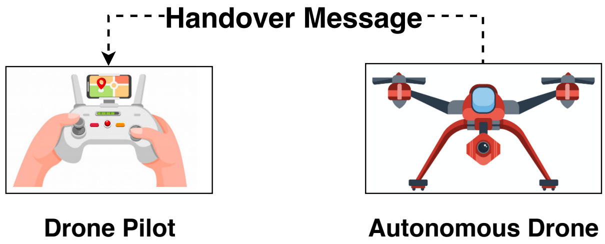

Figure 1: We focus on the drone handover as the main communicative function.

and system stability in various environments and rare critical situations. Second, temporal ontologies can recognize complex drone states by taking into account also events in the past, such as water contact, collisions, or battery states from several minutes ago. Third, many logical reasoners, e.g. ELK [4, 5], Snorocket [6], Sequoia [7] and Metric Temporal Reasoner (MeTeoR) [8] use consequence-based reasoning, which is well-suited to produce formal proofs. One can see these formal proofs as good candidates for explanations of entailments at *inspection time* [9, 10]. At *runtime*, the proof and in particular intermediate derived predicates can be the input for a natural language generation module to produce more flexible and high quality utterances in different environments [11, 12]. Naturally, many sensor data records have numerical values, e.g. temperature, altitude, and battery level. In description logics, concrete domains [13, 14] have been introduced to enable reference to concrete objects (such as numbers) and predefined predicates on these objects (such as numerical comparisons) when defining concepts. Recently, a solution how to incorporate concrete domain reasoning into (atemporal) formal proofs has been proposed [15].

Our long-term goal is to observe and analyze sensor data records over time. Recent work [16] also proposes a tool to answer temporal queries over time-stamped data records with background knowledge, but does not consider explanation services, temporal or concrete domain axioms. In this paper, we focus on reasoning over a temporal ontology, in particular using DatalogMTL [8]. This choice is supported by the fact that the DatalogMTL reasoner MeTeoR[1] [8] is available under an open source license. However, MeTeoR itself supports neither concrete domain reasoning nor proof generation. Thus, we have extended[2] its code to support these two functionalities. Namely, following an approach for extracting description logic (DL) proofs, we implemented proof generation for MeTeoR by tracing internal processes of the system and making the inference steps transparent, and integrated the resulting system with the EVEE library (a tool collection for explaining DL entailments) [17] for finding small proofs for entailed facts. Note that we can only deal with facts that are derived by MeTeoR in the materialization phase, i.e. without the usage of automata, as automata do not naturally fit into a proof-based framework. However, according to the experiments in [8], which are the basis for our evaluation, 99.2% of all facts in a LUBM-based benchmark are indeed derived after finitely many rounds of materialization.

Additionally, we built a DatalogMTL program [18] to describe complex situations that can occur during drone flight. It uses temporal operators as well as concrete domain predicates. Below we provide an example of a critical situation detected by this logic-based framework.

Example 1. *Consider the following DatalogMTL rules with concrete domain predicates.*

$$\boxplus_{[0,\infty)}\mathsf{drone}(\mathsf{X}) :- \mathsf{drone}(\mathsf{X}) \tag{1}$$

$$\mathsf{risk}(\mathsf{X}) :- \mathsf{riskofinternaldamage}(\mathsf{X}) \tag{2}$$

$$\boxplus_{[0,\infty)}\mathsf{riskofinternaldamage}(\mathsf{Y}) :- \mathsf{hightemperature}(\mathsf{Y}),\ \mathsf{drone}(\mathsf{Y}) \tag{3}$$

$$\mathsf{hightemperature}(\mathsf{X}) :- \mathsf{temperature}(\mathsf{X},\mathsf{S}),\ >(\mathsf{S},25) \tag{4}$$

[1] https://pypi.org/project/meteor-reasoner/
[2] https://github.com/de-tu-dresden-inf-lat/meteor-proofs

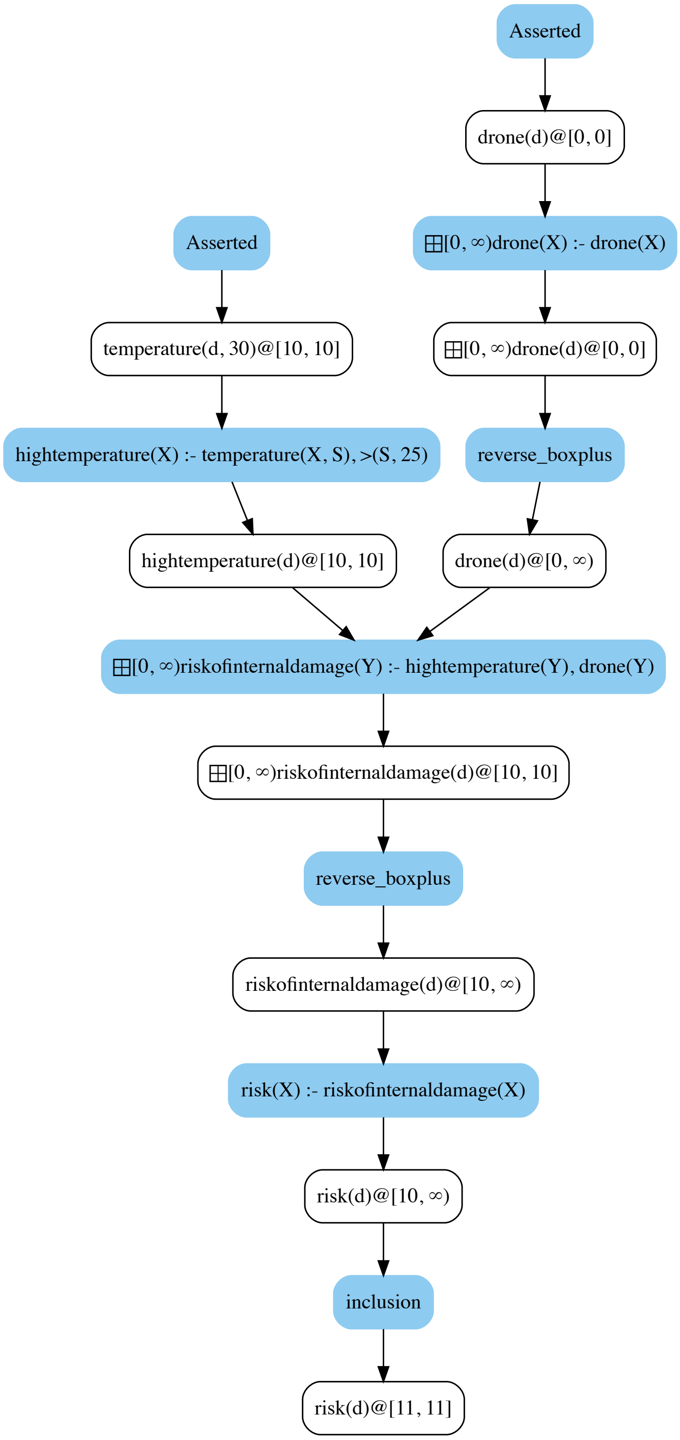

Figure 2: Formal proof generated by our extension of MeTeoR.

They express the following statements: (1) *once an object is a drone, it keeps being a drone forever;* (2) *a risk of internal damage is a risk;* (3) *if a drone experiences a high temperature, then it has a permanent risk of internal damage; and* (4) *the temperature of an object is high if it is above* $25°C$.

Then the dataset $D = \{\mathsf{drone}(\mathsf{d})@0, \mathsf{temperature}(\mathsf{d}, 30)@10\}$ *containing the two timestamped facts that object* d *is a drone at time point* 0*, and at time point* 10 *the temperature of* d *is* $30°C$*, together with the rules above implies that the drone has experienced overheating and thereafter it has a permanent risk of having internal damage. Figure 2 shows a proof for drone* d *still being*

Table 1
Semantics of ground literals [8, Table 1], where $\mathfrak{I}, t \models \top$ holds for all t and $\mathfrak{I}, t \models \bot$ for none.

$\mathfrak{I}, t \models \diamondminus_\varrho A$	iff $\mathfrak{I}, t' \models A$ for some t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models \diamondplus_\varrho A$	iff $\mathfrak{I}, t' \models A$ for some t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models \boxminus_\varrho A$	$\mathfrak{I}, t' \models A$ for all t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models \boxplus_\varrho A$	$\mathfrak{I}, t' \models A$ for all t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models A_1 \mathcal{S}_\varrho A_2$	$\mathfrak{I}, t' \models A_2$ for some t' with $t - t' \in \varrho$ and $\mathfrak{I}, t'' \models A_1$ for all $t'' \in (t', t)$
$\mathfrak{I}, t \models A_1 \mathcal{U}_\varrho A_2$	$\mathfrak{I}, t' \models A_2$ for some t' with $t' - t \in \varrho$ and $\mathfrak{I}, t'' \models A_1$ for all $t'' \in (t, t'')$

at risk at time point 11*, i.e. that* D *entails the fact* $F = \mathsf{risk}(d)@11$.

For the runtime scenario, the intermediate facts involving hightemperature and riskofinternaldamage can be used to create a warning message in natural language.

2. Preliminaries

DatalogMTL is an ontology-based framework for querying temporal data and reasoning over data streams [19, 20]. It extends conventional Datalog rules with metric temporal operators to represent events occurring in the real world at different points in time.

Syntax. In DatalogMTL, an *atom* has the form $P(\tau)$ with P a predicate and τ an n-ary tuple consisting of constants and variables. A *literal* (a.k.a. *metric atom*) A takes one of the following forms, where ϱ is a non-empty positive interval in $\mathbb{Q}$:

$$A \coloneqq \top \mid \bot \mid P(\tau) \mid \diamondminus_\varrho A \mid \diamondplus_\varrho A \mid \boxminus_\varrho A \mid \boxplus_\varrho A \mid A\,\mathcal{S}_\varrho A \mid A\,\mathcal{U}_\varrho A$$

A DatalogMTL *rule* $B \coloneq A_1, \ldots, A_n$ consists of body literals $A_1, \ldots, A_n$, $n \geq 1$, and a head literal B that does not contain the operators $\diamondminus$, $\diamondplus$, $\mathcal{S}$ and $\mathcal{U}$; moreover, all variables in the head must also occur in the body. If an atom, literal or rule contains no variable, it is *ground*. A *fact* F is of the form $A@\varrho$, where A is a ground atom and ϱ a rational interval. Moreover, a finite set of facts is a *dataset*, and a finite set of rules a *program*. If $\varrho_3 = \varrho_1 \cup \varrho_2$ is an interval, then the *coalescing* of the two facts $A@\varrho_1$ and $A@\varrho_2$ is the fact $A@\varrho_3$. The grounding $\mathsf{ground}(\Pi, \mathcal{D})$ of a program Π with respect to a dataset $\mathcal{D}$ is the set of all ground rules obtained by assigning constants from Π and $\mathcal{D}$ to variables in Π.

Semantics. An *interpretation* $\mathfrak{I}$ indicates which ground atoms A hold at which time points $t \in \mathbb{Q}$, written as $\mathfrak{I}, t \models A$. Additionally, if $\mathfrak{I}, t \models A$ for all $t \in \varrho$, we say that $\mathfrak{I}$ *satisfies* the fact $A@\varrho$. The satisfaction relation for other ground literal is shown in Table 1. An interpretation $\mathfrak{I}$ satisfies a ground rule r if, whenever $\mathfrak{I}$ satisfies each body atom of r at a time point t, then $\mathfrak{I}$ also satisfies the head of r at t; it satisfies a (non-ground) rule r if it satisfies each ground instance of r. Finally, $\mathfrak{I}$ is a *model* of a program Π if it satisfies each rule in Π, and it is a model of a dataset $\mathcal{D}$ if it satisfies each fact in $\mathcal{D}$. A program Π and a dataset $\mathcal{D}$ are *consistent* if they have a common model, and they *entail* a fact F, denoted as $\Pi, \mathcal{D} \models F$, if each model of both Π and $\mathcal{D}$ is also a model of F.

Canonical Interpretation. The *immediate consequence operator* T_Π for a program Π maps an interpretation $\mathfrak{I}$ to the least interpretation containing $\mathfrak{I}$ and satisfying the following property for each ground instance r of a rule in Π: whenever $\mathfrak{I}$ satisfies each body atom of r at time point t, then $T_\Pi(\mathfrak{I})$ satisfies the head of r at t. For a dataset $\mathcal{D}$, let $\mathfrak{I}_\mathcal{D}$ be the unique least model of $\mathcal{D}$. The successive application of T_Π to $\mathfrak{I}_\mathcal{D}$ defines a transfinite sequence of interpretations $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D})$ for ordinals α as follows:

1. $T_\Pi^0(\mathfrak{I}_\mathcal{D}) \coloneqq \mathfrak{I}_\mathcal{D}$,
2. $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) \coloneqq T_\Pi(T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D}))$ for α a successor ordinal, and

3. $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) \coloneqq \bigcup_{\beta<\alpha} T_\Pi^\beta(\mathfrak{I}_\mathcal{D})$ for α a limit ordinal.

The *canonical interpretation* $\mathfrak{C}_{\Pi,\mathcal{D}}$ of Π and $\mathcal{D}$ is the interpretation $T_\Pi^{\omega_1}(\mathfrak{I}_\mathcal{D})$, with ω_1 the first uncountable ordinal. If Π and $\mathcal{D}$ have a model, the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ is the least model of Π and $\mathcal{D}$.

MeTeoR. The Metric Temporal Reasoner (MeTeoR) [8] is a scalable reasoner for full DatalogMTL implemented in Python that supports fact entailment via materialization and automata-based reasoning. Materialization (a.k.a. forward chaining) is a classical technique used in conventional Datalog programs. It successively computes consequences of a program Π by applying its rules to a dataset $\mathcal{D}$, similarly to T_Π. Although materialization of regular Datalog programs always terminates, temporal operators can introduce non-terminating behavior, as they may require infinitely many rounds of rule application. To deal with this, a more costly automata-based technique is used in addition to materialization [20]. However, in this paper we consider only facts that are entailed after finitely many materialization rounds, i.e. those that are satisfied by $T_\Pi^n(\mathfrak{I}_\mathcal{D})$ for a finite n. For *bounded* DatalogMTL programs, in which all intervals ρ are bounded, materialization can be used as the basis for a decision procedure for fact entailment [21].

MeTeoR implements the materialization using the Python function `naive_join`[3] that determines all facts that can be derived by applying a given rule r to the current dataset $\mathcal{D}$. To do this, it determines all groundings of the body of r that are satisfied in $\mathfrak{I}_\mathcal{D}$, computes the intersections of the temporal intervals associated to the ground body literals, and adds the resulting head facts to $\mathcal{D}$. To deal with metric body literals, e.g. $\diamondminus_\rho A$, the `apply` function recursively translates all facts of the form $A@\rho'$ in $\mathcal{D}$ into corresponding metric facts $\diamondminus_\rho A@\rho''$ by shifting ρ' according to ρ. Similarly, the `reverse_apply` function uses the reverse calculations to translate a derived metric head fact like $\boxplus_\rho B@\rho'$ into a ground fact $B@\rho''$ that can be added to $\mathcal{D}$. After each materialization round, facts in the extended dataset $\mathcal{D}$ are merged by coalescing, which reduces the space required to store $\mathcal{D}$ and ensures the correctness of `naive_join`.

Proofs and Evee. We consider a consequence $\Pi, \mathcal{D} \models F$ that is to be explained, where Π is a DatalogMTL program, $\mathcal{D}$ a dataset and F a fact. Following [9, 23], we define proofs of $\Pi, \mathcal{D} \models F$ as finite, acyclic, directed hypergraphs, where vertices v are labeled with ground (possibly metric) facts $\ell(v)$ and hyperedges are of the form (S, d), with S a tuple of vertices and d a vertex such that $\{\ell(v) \mid v \in S\} \models \ell(d)$; the leafs of a proof must be labeled by elements of $\mathcal{D}$ and the root by F. In addition, an edge labeling function (see the blue boxes in Figure 2) indicates how the conclusion $\ell(d)$ was derived from the premises, e.g. by applying a DatalogMTL rule or a temporal operator. In general, there can be several proofs for the same entailment. Here, we focus on finding a proof which is minimal according to its size, i.e. the number of vertices. This slightly differs from the proofs for temporal DL query answering in [23], where the root query may contain temporal operators, and ontology axioms are also treated as vertices in the proof.

The Java library Evee[4] (EVincing Expressive Entailments) [17] implements various proof generation methods for different DLs. In particular, Evee can extract size- or depth-minimal proofs using a Dijkstra-like algorithm (e.g. from the output of the $\mathcal{EL}^+$ reasoner Elk) and visualize them like in Figure 2.

3. Tracing Materialization Steps

Given a program Π and a dataset $\mathcal{D}$, our extended version of MeTeoR traces all derived facts and other relevant information throughout the materialization process. Afterward, we use Evee to extract a proof for a target fact F.

Fact Encoding and Tracing. We represent each inference as a simple Python dictionary called `connection`. Each `connection` contains three keys, which we denote as `preds`, `succ`, and `rule`.

[3]We have not yet implemented tracing for the seminaive materialization strategy of MeTeoR [22].
[4]https://github.com/de-tu-dresden-inf-lat/evee

```
{
  "preds": { "alpha": "drone(d)",
             "interval": "[0,0]" },
  "rule": "Boxplus[0,inf)drone(X) :- drone(X)",
  "succ": "Boxplus[0,inf)drone(d)@[0,0]"
}
```

Figure 3: A single `connection` represented as a Python dictionary

Intuitively, `preds` specifies the list of facts from which `succ` is derived, while `rule` represents the inference label. Using native Python dictionaries is simple and fast, whereas initializing custom data structures is often costly. Figure 3 shows the complete encoding of the inference of $\boxplus_{[0,\infty)}\mathsf{drone}(\mathsf{d})@[0,0]$ as seen in Figure 2. The global hypergraph $\mathcal{G}$ collects each `connection` produced during the various stages of materialization. In addition to the DatalogMTL rules themselves, we also trace the application of temporal operators in the `apply` and `reverse_apply` functions.

Reverse Apply. For the `reverse_apply` function in particular, we had to do more extensive modifications. In its original form, the source code was not suitable to connect a metric fact to an underlying non-metric fact, in particular for multiple nested operators, e.g. as in $\boxminus_\rho \diamondplus_{\rho'} A$, since the metric operators were processed in a different order. Thus, we adjusted the existing `reverse_apply` function to enable step-wise recording of complex facts in an outside-inside fashion. First, the metric fact is passed to `reverse_apply` along with an auxiliary list $\mathcal{L}$ for recording relevant information. Second, its outermost temporal operator is recursively popped and the interval corresponding to the inner literal is recorded in $\mathcal{L}$. Last, we stop the tracing process when the innermost atom and its interval are reached, and merge $\mathcal{L}$ with $\mathcal{G}$ by converting each traced fact into a `connection` as described above.

Pre-processing, Coalescing and Post-processing. Since the initial dataset is required to construct a full proof with Evee, each fact from $\mathcal{D}$ is already added to $\mathcal{G}$ with label "Asserted" (see Figure 2) while loading the dataset. Furthermore, as mentioned in Section 2, time intervals of ground facts are coalesced to represent the final dataset in a more human-readable way and to ensure the correctness of the reasoner. Therefore, we extend MeTeoR accordingly to trace the coalescing step whenever it occurs in the reasoning process and record the information in $\mathcal{G}$. After MeTeoR is finished, we run a simple conversion step to transform the inferences recorded in $\mathcal{G}$ into a JSON format compatible with Evee. Finally, a small Java program using Evee reads the JSON file and outputs a size-minimal proof for the goal fact F.

4. Tracing Evaluation

We conducted a small evaluation of the feasibility of the described approach.

Machine Configuration. We used a desktop machine with Windows 10 Home Edition (64-bit), AMD Ryzen 5600 3.5 GHz CPU, 32 GB of RAM, and memory channels operating on 3200 MHz. All results are produced from a single attempt.

LUBM Benchmark. We used the LUBM benchmark generator [24] and scripts from [8] to construct several temporal datasets, ranging in size from 10 k to 100 k facts. Each fact is randomly assigned a time interval. We use the same DatalogMTL program as in [8], which contains 85 rules over the LUBM predicates. Due to randomization in the dataset generation, we cannot use the same facts as in the experiments of [8]. Instead, we run the materialization for 6 rounds, after which the majority of rules have been applied and only a few recursive rules can be applied indefinitely. We then chose facts derived in the last materialization round to

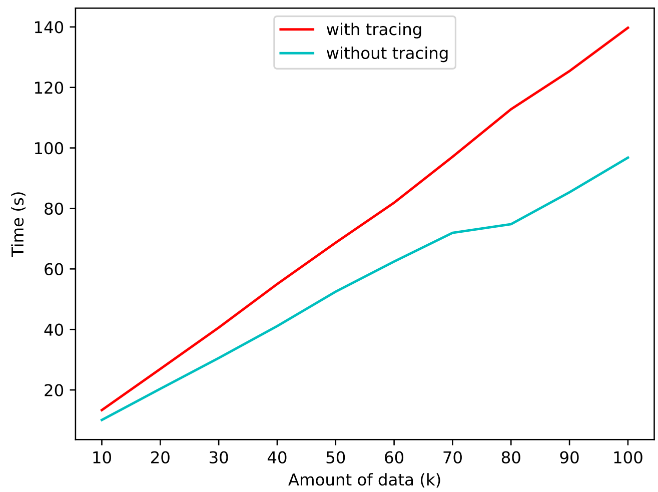

Figure 4: CPU time comparison between enabled and disabled tracing in MeTeoR

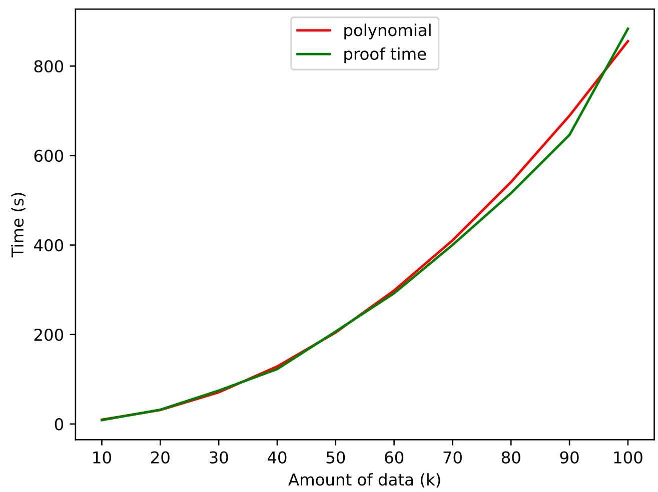

Figure 5: Runtime of proof extraction with Evee, compared to a quadratic polynomial

generate proofs.

Tracing in MeTeoR. We hypothesize that, while tracing slows down the reasoning process of MeTeoR, it does not significantly worsen the overall performance. To test this, we compare the total CPU time of our implementation for computing the hypergraph $\mathcal{G}$ to the runtime of the unmodified reasoner, and report the results in Figure 4. As the size of the dataset grows, a more significant performance gap is seen between the baseline and our implementation. However, on average, there is only a 37% increase of runtime, and the tracing does not destroy the nice linear scaling behavior of MeTeoR, which confirms our hypothesis.

Proof Extraction. We also measure the runtime of the Evee library for finding a minimal proof in $\mathcal{G}$, with the results shown in Figure 5. Compared to the tracing, Evee takes considerably longer to produce minimal-size proofs (e.g. 884 s vs. 140 s on the largest dataset). The reason behind this blow-up is the polynomial time complexity of finding minimal proofs [10]. In particular, the Dijkstra-like algorithm requires nested iteration over all inferences deriving a specific fact, in order to find the ones resulting in minimal proofs. Figure 5 also shows a quadratic function fitted on the first 5 data points (in red), which differs only minimally from the real runtimes. Consequently, this shows that not only is the proof extraction with EVEE theoretically tractable, but it is also efficient. For the real-time drone use case, we expect much fewer than 10k input facts, since we only have to consider measurements of around 20–40 sensors over the last 20–60 seconds.

5. Proof Verbalization

The overall goal of the larger system is to convey critical information during handover and enable the drone operator to take control. Such critical information may be given as an auditory message, as the operator will already be dealing with lots of other visual input. The generated temporal proofs are represented as a Python dictionary or in JSON (see Figure 3) and contain a long step-by-step logical proof. This makes it hard to present relevant information from proofs comprehensibly to drone pilots.

Recently emerging large language models (LLMs) such as GPT-4 [25], LLaMA-2 [26], Gemini [27] can be prompted to perform a wide variety of generative tasks, including code completion and documentation generation. This is ideal for our setup, because prompting strategies such as few-shot prompting only need a few input examples, and do not require a large in-domain dataset for training. We use Gemini-Pro [27] LLM for our experiments, as it performs better than GPT-3.5-Turbo on particularly long and complex reasoning tasks [28]. We deploy Gemini in a zero-shot manner to generate an actionable explanation using temporal proofs.

Approach. Generating explanations from proofs requires an LLM to identify the relevant proof steps that provide reasoning for the critical situation and verbalize these steps in a coherent manner. We first attempt a select-then-summarize prompting approach, where we prompt the LLM to filter relevant proof steps and then verbalize them. Although the LLM could identify relevant proof steps, it could not generate a coherent explanation only from the selected steps. We speculate that this was due to the missing context of the entire situation. Hence, we propose a verbalize-then-summarize (VeSum) prompting approach for generating explanations from the

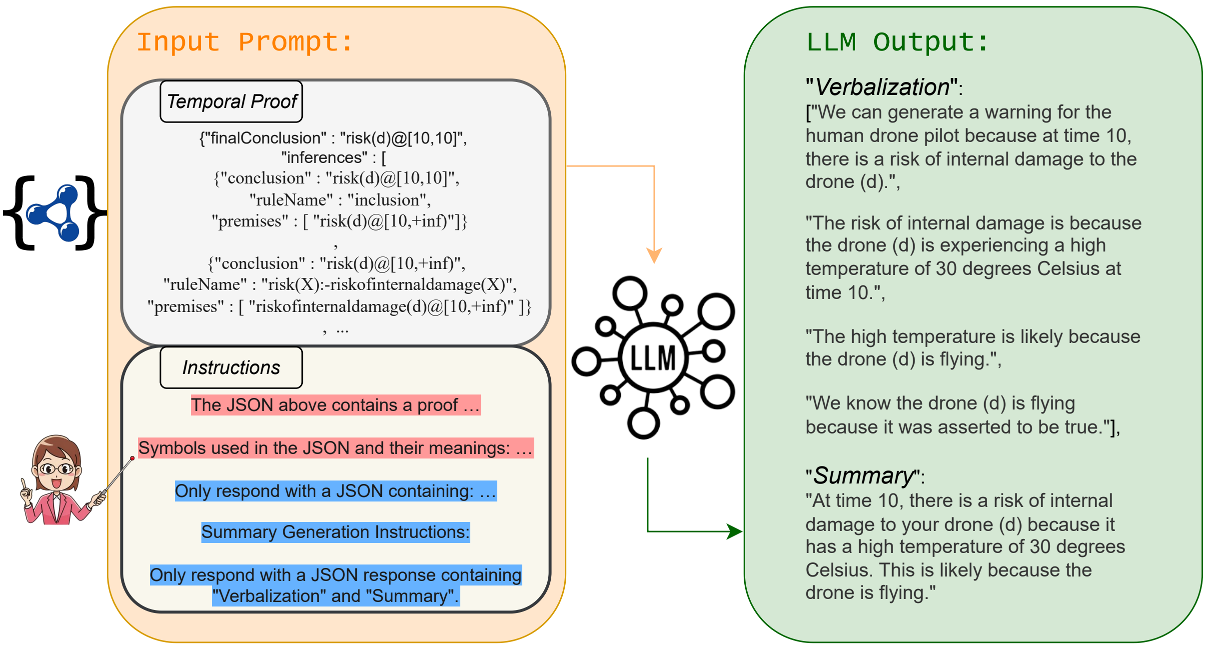

Figure 6: VeSum prompt design and a verbalization example using Gemini-Pro. The input prompt consists of Temporal Proof and Instructions that guide LLMs in generating explanations. The instructions are divided into two parts – data description instructions and task-specific instructions.

proofs. VeSum first instructs the LLM to produce a verbalization of the entire proof, and then use the generated verbalization to generate an explanation summary. VeSum is inspired by well-known prompting approaches – Chain-of-Thought (CoT) [29] and ReAct [30] prompting. The prompt design of VeSum and the sample response for the example input (see Figure 2) are shown in Figure 6.

Challenges. The major challenges of using LLMs for generating explanations from proof trees are as follows:

- LLMs are very sensitive to input instructions [31], and input templates [32]. Hence, finding the optimal instructions for long text generation is challenging.
- For our task, the predicate names are mostly descriptive and verbose (e.g. $\mathsf{hightemperature}(\mathsf{X}) \coloneq \mathsf{temperature}(\mathsf{X},\mathsf{S}), {>}(\mathsf{S},25)$). In our preliminary experiments, when we shortened the names (e.g. $\mathsf{hightemperature}(\mathsf{X})$ to $\mathsf{HT}(\mathsf{X})$), we found that models failed to understand the meaning of the rules. This suggests that the predicate names have a noticeable impact on the generated explanations, hence they should be chosen carefully. Alternatively, one could provide additional descriptions of each predicate via string annotations and incorporate them into the prompt.
- LLMs can write fluently and grammatically; their advantage over traditional template-based methods lies in their generalizability to new sensor input and better naturalness and variability of expression. However, the generated text is not always grounded in the given information. For instance, if we take a closer look at the generated summary in Figure 6, we can see that LLM generates the reason for the warning as "because it has a high temperature of 30 degrees Celsius. This is likely because the drone is flying." While the proof mentions the high temperature, it does not specify any further reason for this measurement (see Figure 2).

6. Conclusions and Future Work

Currently, our MeTeoR-based system can explain facts entailed by a fixed dataset that involves timestamps and numerical measurements. To achieve this, we implemented inference tracing for MeTeoR and used EVEE to extract small proofs. We showed that the additional components have a minor impact on computational resources. We have started to explore how to generate natural language explanations from these proofs.

In the future, we want to improve our system in several directions: (i) consider real-time operations in a streaming setting, in which new facts continuously come in and old facts are removed, (ii) improve the quality of the generated verbal messages, and (iii) reduce cognitive load for the drone operator in situations with multiple warnings.

For (i), assuming that sensor measurements arrive incrementally every second, we should avoid re-generating the same proof over and over again if it still holds. Using the streaming version of MeTeoR for forward-propagating rules [33], our system could also be adapted to the forgetting mechanism using sliding windows. This dramatically reduces the data size and makes real-time processing feasible. Nevertheless, we can still improve the tracing and proof extraction themselves, e.g. by goal-directed tracing of the inferences that lead to a target fact, or by using optimized data structures for proof extraction.

For (ii), we will first investigate better prompting techniques that leverage proofs to generate more strongly grounded explanations. We will also experiment with more diverse proof formats based on first-order logic or description logic to test the models' abilities. We also aim to develop a systematic understanding of how different phrasing choices impact explanation quality, which will inform guidelines for designing new ontologies and improving prompting methods for existing ones. Additionally, we will explore the latest open-source LLMs, evaluating their potential for fine-tuning on the annotated data to improve task performance.

For (iii), we must identify which situations are more critical than others, and should be delivered to the drone operator with the highest priority. Here, the system should be able to rank warnings and their explanations according to their severity, and also convey the severity level verbally.

Acknowledgements This work was partially supported by the DFG in grant 389792660 as part of TRR 248 (https://perspicuous-computing.science).

References

[1] T. Fuhrman, D. Schneider, F. Altenberg, T. Nguyen, S. Blasen, S. Constantin, A. Waibe, An interactive indoor drone assistant, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 6052–6057. doi:10.1109/IROS40897.2019.8967587.

[2] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017. doi:10.1017/9781139025355.

[3] C. Baral, M. Gelfond, Logic programming and knowledge representation, The Journal of Logic Programming 19-20 (1994) 73–148. doi:10.1016/0743-1066(94)90025-6, Special Issue: Ten Years of Logic Programming.

[4] Y. Kazakov, M. Krötzsch, F. Simancik, The incredible ELK – from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies, J. Autom. Reasoning 53 (2014) 1–61. doi:10.1007/s10817-013-9296-3.

[5] Y. Kazakov, P. Klinov, A. Stupnikov, Towards reusable explanation services in Protege, in: A. Artale, B. Glimm, R. Kontchakov (Eds.), Proc. of the 30th Int. Workshop on Description Logics (DL'17), volume 1879 of *CEUR Workshop Proceedings*, 2017. URL: http://www.ceur-ws.org/Vol-1879/paper31.pdf.

[6] A. Metke-Jimenez, M. Lawley, Snorocket 2.0: Concrete domains and concurrent classification, in: S. Bail, B. Glimm, R. S. Gonçalves, E. Jiménez-Ruiz, Y. Kazakov, N. Matentzoglu, B. Parsia (Eds.), Informal Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE-2013), volume 1015 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2013, pp. 32–38. URL: http://ceur-ws.org/Vol-1015/paper_3.pdf.

[7] A. Bate, B. Motik, B. C. Grau, F. Simancik, I. Horrocks, Extending consequence-based reasoning to SRIQ, in: C. Baral, J. P. Delgrande, F. Wolter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR, AAAI Press, 2016, pp. 187–196. URL: http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12882.

[8] D. Wang, P. Hu, P. A. Walega, B. C. Grau, MeTeoR: Practical reasoning in Datalog with metric temporal operators, in: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, AAAI Press, 2022, pp. 5906–5913. doi:10.1609/AAAI.V36I5.20535.

[9] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann, A. Kovtunova, Finding small proofs for description logic entailments: Theory and practice, in: E. Albert, L. Kovacs (Eds.), LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, volume 73 of *EPiC Series in Computing*, EasyChair, 2020, pp. 32–67. doi:10.29007/nhpp.

[10] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann, A. Kovtunova, Finding good proofs for description logic entailments using recursive quality measures, in: A. Platzer, G. Sutcliffe (Eds.), Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Proceedings, volume 12699 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 291–308. doi:10.1007/978-3-030-79876-5_17.

[11] S. Borgwardt, E. Chang, K. Chapman, V. Demberg, A. Kovtunova, H. Yeh, Logic-guided neural utterance generation from drone sensory data (extended abstract), in: M. Homola,

V. Ryzhikov, R. A. Schmidt (Eds.), Proceedings of the 34th International Workshop on Description Logics (DL 2021), volume 2954 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: https://ceur-ws.org/Vol-2954/abstract-11.pdf.

[12] E. Chang, A. Kovtunova, S. Borgwardt, V. Demberg, K. Chapman, H. Yeh, Logic-guided message generation from raw real-time sensor data, in: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, S. Piperidis (Eds.), Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, European Language Resources Association, 2022, pp. 6899–6908. URL: https://aclanthology.org/2022.lrec-1.745.

[13] F. Baader, P. Hanschke, A scheme for integrating concrete domains into concept languages, in: J. Mylopoulos, R. Reiter (Eds.), Proc. IJCAI Conference, Morgan Kaufmann, 1991, pp. 452–457. URL: http://ijcai.org/Proceedings/91-1/Papers/070.pdf.

[14] C. Lutz, Description logics with concrete domains—a survey, in: Advances in Modal Logics Volume 4, World Scientific Publishing Co. Pte. Ltd., 2003.

[15] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann, A. Kovtunova, Combining proofs for description logic and concrete domain reasoning, in: A. Fensel, A. Ozaki, D. Roman, A. Soylu (Eds.), Rules and Reasoning - 7th International Joint Conference, RuleML+RR 2023, Proceedings, volume 14244 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 54–69. doi:10.1007/978-3-031-45072-3_4.

[16] L. Westhofen, C. Neurohr, J. C. Jung, D. Neider, Answering temporal conjunctive queries over description logic ontologies for situation recognition in complex operational domains, in: B. Finkbeiner, L. Kovács (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 30th International Conference, TACAS, volume 14570 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 167–187. doi:10.1007/978-3-031-57246-3_10.

[17] C. Alrabbaa, S. Borgwardt, T. Friese, P. Koopmann, J. Méndez, A. Popovič, On the eve of true explainability for OWL ontologies: Description logic proofs with Evee and Evonne, in: O. Arieli, M. Homola, J. C. Jung, M. Mugnier (Eds.), Proceedings of the 35th International Workshop on Description Logics (DL 2022), volume 3263 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3263/paper-2.pdf.

[18] A. Kovtunova, S. Borgwardt, Drone program for critical situation detection over sensor data streams, 2024. doi:10.5281/zenodo.10822865.

[19] S. Brandt, E. G. Kalayci, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Querying log data with metric temporal logic, J. Artif. Intell. Res. 62 (2018) 829–877. doi:10.1613/jair.1.11229.

[20] P. A. Walega, B. C. Grau, M. Kaminski, E. V. Kostylev, DatalogMTL: Computational complexity and expressive power, in: S. Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI, ijcai.org, 2019, pp. 1886–1892. doi:10.24963/ijcai.2019/261.

[21] P. A. Walega, M. Zawidzki, D. Wang, B. C. Grau, Materialisation-based reasoning in DatalogMTL with bounded intervals, in: B. Williams, Y. Chen, J. Neville (Eds.), Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2023, pp. 6566–6574. doi:10.1609/AAAI.V37I5.25807.

[22] D. Wang, P. A. Walega, B. C. Grau, Seminaïve materialisation in DatalogMTL, in: G. Governatori, A. Turhan (Eds.), Rules and Reasoning - 6th International Joint Conference on Rules and Reasoning, RuleML+RR, volume 13752 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 183–197. doi:10.1007/978-3-031-21541-4_12.

[23] C. Alrabbaa, S. Borgwardt, P. Koopmann, A. Kovtunova, Finding good proofs for answers to conjunctive queries mediated by lightweight ontologies, in: O. Arieli, M. Homola, J. C. Jung, M. Mugnier (Eds.), Proceedings of the 35th International Workshop on Description Logics (DL 2022), volume 3263 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3263/paper-3.pdf.

[24] Y. Guo, Z. Pan, J. Heflin, LUBM: A benchmark for OWL knowledge base systems, J. Web Semant. 3 (2005) 158–182. doi:10.1016/j.websem.2005.06.005.

[25] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., GPT-4 technical report, arXiv preprint arXiv:2303.08774 (2023). doi:10.48550/arXiv.2303.08774.

[26] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: Open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288 (2023). doi:10.48550/arXiv.2307.09288.

[27] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al., Gemini: A family of highly capable multimodal models, arXiv preprint arXiv:2312.11805 (2023). doi:10.48550/arXiv.2312.11805.

[28] S. N. Akter, Z. Yu, A. Muhamed, T. Ou, A. Bäuerle, Á. A. Cabrera, K. Dholakia, C. Xiong, G. Neubig, An in-depth look at Gemini's language abilities, arXiv preprint arXiv:2312.11444 (2023). doi:10.48550/arXiv.2312.11444.

[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Advances in neural information processing systems 35 (2022) 24824–24837. URL: https://openreview.net/forum?id=_VjQlMeSB_J.

[30] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, Y. Cao, ReAct: Synergizing reasoning and acting in language models, in: The Eleventh International Conference on Learning Representations, 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X.

[31] M. Loya, D. Sinha, R. Futrell, Exploring the sensitivity of LLMs' decision-making capabilities: Insights from prompt variations and hyperparameters, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 3711–3716. doi:10.18653/v1/2023.findings-emnlp.241.

[32] M. Sclar, Y. Choi, Y. Tsvetkov, A. Suhr, Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting, arXiv preprint arXiv:2310.11324 (2023). doi:10.48550/arXiv.2310.11324.

[33] P. A. Walega, M. Kaminski, D. Wang, B. C. Grau, Stream reasoning with DatalogMTL, J. Web Semant. 76 (2023) 100776. doi:10.1016/j.websem.2023.100776.