Generating Activity Definitions with Large Language Models

Andreas Kouvaras
andreas@unipi.gr
University of Piraeus, GR
VesselFront, GR

Periklis Mantenoglou
pmantenoglou@iit.demokritos.gr
NCSR “Demokritos”, GR

Alexander Artikis
a.artikis@unipi.gr
University of Piraeus, GR
NCSR “Demokritos”, GR

ABSTRACT

Composite activity recognition systems reason over streams of low-level, symbolic events in order to detect instances of composite activities, based on their formal definitions. Composite activity definition construction is a highly involved task, as these definitions often involve numerous spatio-temporal constraints, while labels that may be used for learning them automatically are hard to obtain. To address this issue, we propose a method that generates composite activity definitions from natural language descriptions using pre-trained Large Language Models (LLMs). In order to assess the quality of LLM-generated definitions, we propose a novel similarity metric, which reflects the human effort required to correct them. We present a thorough experimental evaluation of our approach on the maritime domain, demonstrating its effectiveness.

1 INTRODUCTION

Composite event recognition (CER) involves the detection of composite activities by means of temporal pattern matching over streams of low-level, symbolic events [2, 17, 18, 38]. CER requires a formal language, determining the syntax of the temporal patterns defining composite activities. Consider, e.g., maritime situational awareness, where the CER task is to report dangerous, illegal and suspicious vessel activities with minimal latency [5]. In order to detect illegal fishing, we may use a pattern specifying that a vessel performs several consecutive turns while sailing in an environmentally protected area at a speed that is typical for fishing. Constructing such a pattern is an involved task, as we need to combine several simpler vessel activities, i.e., entering an area and making a turn, together with spatial information, i.e., the locations of protected areas, and background knowledge, i.e., the typical velocity of a fishing vessel.

The construction of definitions for composite activities, such as illegal fishing, introduces further challenges. Composite activity definitions should be specified in a formal language with unambiguous semantics; formalising a natural language description of an activity definition may be an arduous and error-prone task that demands specialised programming skills. Furthermore, methods that learn composite activity definitions, such as [16, 22, 28, 29], require big datasets for training, including a (partial) labeling of composite activity instances. Due to the inherent infrequency of composite activities, as well as privacy and security concerns regarding the dissemination of such data in sensitive domains (consider, e.g., the cases of illegal activities in the maritime domain), labels of composite activity instances are mostly unavailable, prohibiting the use of such learning techniques.

To address these issues, we propose a method that generates composite activity definitions from natural language descriptions using pre-trained Large Language Models (LLMs). LLMs have proven effective semantic parsers, transforming natural language descriptions into a formal form [19, 20, 32, 37, 39]. We generate composite activity definitions in the language of the “Run-Time Event Calculus” (RTEC)[1], which represents such definitions with logic programming rules [4, 27, 34]. Contrary to automata-based approaches for CER, such as [1, 9, 18, 40], RTEC supports definitions of relational composite activities with background knowledge. Unlike other logic-based frameworks, such as [7, 8, 13, 35, 36], RTEC supports durative composite activities that may be subject to the common-sense law of inertia, such as illegal fishing, as well as activity hierarchies that pave the way for caching. RTEC is an implementation of the Event Calculus, i.e., a logic programming formalism for representing events and reasoning about their effects over time [24]. Contrary to other Event Calculus-based frameworks [3, 6, 10, 14, 21, 30], RTEC is optimised for CER by means of windowing and caching techniques, and has proven highly efficient in real applications, including maritime situational awareness [33] and commercial fleet management [34], outperforming competing systems [26, 27].

The reasoning abilities of LLMs, particularly in multi-step logical reasoning tasks, remain constrained [12, 19, 20]. Such reasoning tasks become even more demanding when we need to reason over sequences of events that evolve over time, like, e.g., streams of vessel position signals. These shortcomings aside, LLMs are increasingly able to generate formal language expressions, such as SQL queries [15] and Answer Set Programming (ASP) specifications [20], based on natural language descriptions of these expressions. Ishay et al. utilised LLMs with prompt engineering to obtain ASP solutions for logic puzzles [20]. Coppolillo et al. fine-tuned lightweight LLMs for optimising ASP performance in specific tasks [12]. ChatLogic incorporates iterative correction modules to refine the generated code in order to enhance multi-step logical reasoning [37]. ‘Chain-of-thought’ prompting often improves the interpretability and consistency of LLM-generated logic specifications [12, 23, 37].

Unlike the aforementioned approaches, we propose a prompting approach that enables LLMs to construct composite activity definitions in the language of RTEC, which may be subsequently used for reasoning over data streams, in order to detect complex spatio-temporal phenomena of interest. Since LLMs are statistical models, the definitions generated by this approach may include errors, in the form of, e.g., syntactic mistakes or conditions with undefined activities. To tackle this issue, we introduce an automated quantitative evaluation technique, assessing the quality of LLM-generated definitions using a novel similarity metric that reflects the human effort required to correct these definitions. Thus, our approach extends qualitative error analyses [20, 39], which rely solely on subjective interpretations.

Our contributions may then be summarised as follows. First, we present a prompting method for constructing composite activity definitions in the language of RTEC using pre-trained LLMs. Second, we propose a framework that evaluates the quality of the

[1]https://github.com/aartikis/rtec

LLM-generated composite activity definitions using a novel similarity metric. Third, we present a thorough empirical analysis on maritime situational awareness using a wide range of LLMs and real maritime data. The results show that our proposed approach produces activity definitions that may be successfully used in practice, as these achieve high predictive accuracy.

2 BACKGROUND: RTEC

RTEC is a formal, logic programming framework that extends the Event Calculus with optimisation techniques for CER [4, 26, 27].

Syntax. The language of RTEC includes sorts for representing time, instantaneous events and fluents, i.e., properties whose values may change over time. RTEC employs a linear time-line with non-negative integer time-points. A 'fluent-value pair' (FVP) $F{=}V$ denotes that fluent F has value V. $\mathsf{happensAt}(E, T)$ signifies that event E occurs at time-point T. $\mathsf{initiatedAt}(F = V, T)$ (resp. $\mathsf{terminatedAt}(F = V, T)$) expresses that a time period during which a fluent F has the value V continuously is initiated (terminated) at T. $\mathsf{holdsAt}(F = V, T)$ states that F has value V at T, while $\mathsf{holdsFor}(F = V, I)$ expresses that $F{=}V$ holds continuously in the intervals included in list I.

A formalisation of the activity definitions of a domain in RTEC is called *event description*. An event description may contain rules defining two types of FVPs: 'simple' and 'statically determined'. A simple FVP is defined using a set of initiatedAt and terminatedAt rules, and is subject to the commonsense law of inertia, i.e., an FVP $F{=}V$ holds at a time-point T, if $F{=}V$ has been 'initiated' by an event at a time-point earlier than T, and not 'terminated' by another event in the meantime.

Example 2.1 (Within area). In maritime monitoring, an activity may be disallowed in certain areas, e.g., fisheries restricted areas. Thus, it is desirable to compute the intervals during which a vessel is in such an area. See the definition of a simple FVP below:

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true},\ T) \leftarrow\\ &\quad \mathsf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID}), T),\\ &\quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{aligned} \tag{1}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true},\ T) \leftarrow\\ &\quad \mathsf{happensAt}(\mathit{leavesArea}(\mathit{Vl}, \mathit{AreaID}), T),\\ &\quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{aligned} \tag{2}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true},\ T) \leftarrow\\ &\quad \mathsf{happensAt}(\mathit{gapStart}(\mathit{Vl}), T).\end{aligned} \tag{3}$$

withinArea(*Vl*, *AreaType*) is a Boolean fluent denoting that a vessel *Vl* is in an area of type *AreaType*, while *entersArea*(*Vl*, *AreaID*), *leavesArea*(*Vl*, *AreaID*) and *gapStart*(*Vl*) are input events, derived by the online processing of vessel position signals, and their spatial relations with areas of interest. *areaType*(*AreaID*, *AreaType*) is an atemporal predicate storing background knowledge regarding the types of areas in a dataset. Rules (1) and (2) state that *withinArea*(*Vl*, *AreaType*) is initiated (resp. terminated) as soon as vessel *Vl* enters (leaves) an area *AreaID*, whose type is *AreaType*. Rule (3) expresses that *withinArea*(*Vl*, *AreaType*) is terminated when there is a communication gap, i.e., when *Vl* stops transmitting its position, and we become uncertain of its whereabouts. ◊

Definition 2.2 (Syntax of Rules Defining Simple FVPs). Consider a simple FVP $F = V$. The $\mathsf{initiatedAt}(F = V, T)$ rules of the event description have the following syntax:

$$\begin{aligned}&\mathsf{initiatedAt}(F = V,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(E_1,\ T)[[, [\mathsf{not}]\ \mathsf{happensAt}(E_2,\ T),\ \ldots,\\ &\quad [\mathsf{not}]\ \mathsf{happensAt}(E_n,\ T), [\mathsf{not}]\ \mathsf{holdsAt}(F_1 = V_1,\ T),\ \ldots,\\ &\quad [\mathsf{not}]\ \mathsf{holdsAt}(F_k = V_k,\ T)]].\end{aligned}$$

The first body literal of an initiatedAt rule is a positive happensAt predicate; this is followed by a possibly empty set, denoted by '[[ ]]', of positive/negative happensAt and holdsAt predicates. 'not' expresses negation-by-failure [11], while '[not]' denotes that 'not' is optional. All (head and body) predicates are evaluated on the same time-point T. The bodies of $\mathsf{terminatedAt}(F = V, T)$ rules have the same form. ■

A statically determined FVP $F = V$ is defined via a rule with head $\mathsf{holdsFor}(F = V, I)$. This rule computes the maximal intervals during which $F = V$ holds continuously by applying a set interval manipulation operations, i.e., union_all, intersect_all and relative_complement_all, on the maximal intervals of other FVPs.

Example 2.3 (Anchored and moored vessels). Consider the following definition of a statically determined FVP:

$$\begin{aligned}&\mathsf{holdsFor}(\mathit{anchoredOrMoored}(\mathit{Vl}) = \mathsf{true}, I) \leftarrow\\ &\quad \mathsf{holdsFor}(\mathit{stopped}(\mathit{Vl}) = \mathit{farFromPorts}, I_{sf}),\\ &\quad \mathsf{holdsFor}(\mathit{withinArea}(\mathit{Vl}, \mathit{anchorage}) = \mathsf{true}, I_a),\\ &\quad \mathsf{intersect\_all}([I_{sf}, I_a], I_{sfa}),\\ &\quad \mathsf{holdsFor}(\mathit{stopped}(\mathit{Vl}) = \mathit{nearPorts}, I_{sn}),\\ &\quad \mathsf{union\_all}([I_{sfa}, I_{sn}], I).\end{aligned} \tag{4}$$

anchoredOrMoored(*Vl*) is a Boolean statically determined fluent, defined in terms of three other FVPs: *stopped*(*Vl*) = *farFromPorts*, *stopped*(*Vl*) = *nearPorts* and *withinArea*(*Vl*, *anchorage*) = true. The multi-valued fluent *stopped*(*Vl*) expresses the periods during which vessel *Vl* is idle near some port or far from all ports. Rule (4) derives the intervals during which vessel *Vl* is both stopped far from all ports and within an anchorage area, by applying the intersect_all operation on the lists of maximal intervals I_{sf} and I_a. The output of this operation is list I_{sfa}. Subsequently, list I is derived by applying union_all on lists I_{sfa} and I_{sn}. In this way, list I contains the maximal intervals during which vessel *Vl* has stopped near some port or within an anchorage area. ◊

Definition 2.4 (Syntax of Rules Defining Statically Determined FVPs). The definition of statically determined FVP $F = V$ is a rule that has the following syntax:

$$\begin{aligned}&\mathsf{holdsFor}(F = V,\ I_{n+m}) \leftarrow\\ &\quad \mathsf{holdsFor}(F_1 = V_1,\ I_1)[[, \mathsf{holdsFor}(F_2 = V_2,\ I_2),\ \ldots\\ &\quad \mathsf{holdsFor}(F_n = V_n,\ I_n), \mathsf{intervalConstruct}(L_1,\ I_{n+1}),\ \ldots\\ &\quad \mathsf{intervalConstruct}(L_m,\ I_{n+m})]].\end{aligned}$$

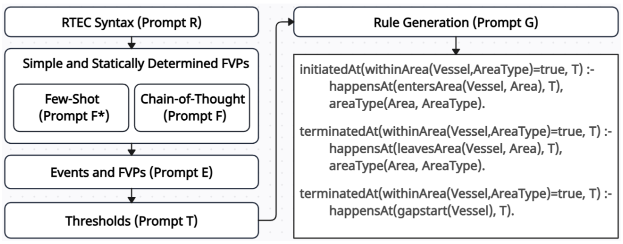
The first body literal of a holdsFor rule defining $F = V$ is a holdsFor predicate expressing the maximal intervals of an FVP other than $F = V$. This is followed by a possibly empty list, denoted by '[[ ]]', of holdsFor predicates and interval manipulation constructs, expressed by intervalConstruct. $\mathsf{intervalConstruct}(L_j, I_{n+j})$ may be one of the following: $\mathsf{union\_all}(L_j, I_{n+j})$, $\mathsf{intersect\_all}(L_j, I_{n+j})$ or $\mathsf{relative\_complement\_all}(I_k, L_j, I_{n+j})$. I_k, where $k < n + j$, is a list of maximal intervals appearing earlier in the body of the rule, and list L_j contains a subset of these lists. The output list I_{n+m} contains the maximal intervals during which $F = V$ holds continuously. ■

Examples 2.1 and 2.3 illustrate that simple and statically determined FVPs may be used to express composite activities. Based on the syntax of rules with head holdsFor (see, e.g., rule (4)), a statically determined FVP holds as long as a Boolean combination of other FVPs is satisfied. Thus, statically determined FVPs are tailored for modeling composite activities that are defined based on other activities by means of conjunction, disjunction and negation operators, such as ‘anchored or moored’. Apart from these types of activity, RTEC may also express ‘inertial’ composite activities, i.e., activities that persist through time and may only come about (or conclude) based on the satisfaction of a set of instantaneous conditions. Such conditions may be expressed using a set of initiatedAt and terminatedAt rules, effectively defining a simple FVP (see, e.g., rules (1)–(3)).

Reasoning. The key reasoning task of RTEC is the computation of holdsFor$(F=V, I)$, i.e., the list of maximal intervals I during which each FVP holds continuously. For a simple FVP $F=V$, RTEC first computes the initiations and the terminations of $F=V$, by evaluating its initiatedAt and its terminatedAt rules, respectively. Next, RTEC computes the maximal intervals of $F=V$ by matching each initiation T_s of $F{=}V$ with the first termination T_e of $F{=}V$ after T_s, ignoring every intermediate initiation between T_s and T_e. RTEC may then derive holdsAt$(F=V, T)$ by checking whether T belongs to one of the maximal intervals of $F{=}V$. In the case of a statically determined FVP $F=V$, RTEC computes holdsFor$(F=V, I)$ by evaluating the conditions of the holdsFor rule with FVP $F=V$ in its head.

RTEC supports hierarchical event descriptions, where it is possible to compute and cache the maximal intervals of FVPs in a bottom-up manner. At each ‘query time’ q_i, RTEC takes into consideration the events that fall within a specified sliding window with size ω. All events that took place before or at $q_i{-}\omega$ are discarded/‘forgotten’. This way, the cost of reasoning depends on ω, instead of the size of the complete stream.

3 ACTIVITY DEFINITION GENERATION


RTEC Syntax (Prompt R)
Simple and Statically Determined FVPs
Few-Shot (Prompt F*)
Chain-of-Thought (Prompt F)
Events and FVPs (Prompt E)
Thresholds (Prompt T)
Rule Generation (Prompt G)

```
initiatedAt(withinArea(Vessel,AreaType)=true, T) :-
    happensAt(entersArea(Vessel, Area), T),
    areaType(Area, AreaType).

terminatedAt(withinArea(Vessel,AreaType)=true, T) :-
    happensAt(leavesArea(Vessel, Area), T),
    areaType(Area, AreaType).

terminatedAt(withinArea(Vessel,AreaType)=true, T) :-
    happensAt(gapstart(Vessel), T).
```

Figure 1: LLM prompting for composite activity definition generation.

Figure 1 illustrates our prompting approach for generating composite maritime activities in the language of RTEC. Similar to how a human would approach the task, we start by teaching the LLM the syntax of the language, i.e., the main predicates of RTEC — see "RTEC syntax (Prompt R)" in Figure 1. Then, we proceed with the distinction between simple and statically determined FVPs — recall that these are the possible ways in which a composite activity may by expressed. For this step, we provide two alternative routes: "few-shot" prompting (prompt F*), as well as "chain-of-thought" prompting (prompt F). In our empirical analysis we found that zero-shot prompting produced poor results, and thus we do not include it in our pipeline. Third, we present to the LLM the items of the input stream, i.e., the syntax of the input events and FVPs (prompt E). These events and FVPs may be used in the bodies of the rules expressing composite maritime activities. Fourth, we present further domain-specific information; for maritime situational awareness, there are several threshold values that should be taken into consideration, concerning, among others, the expected sailing speed per vessel type and the allowed sailing speed near the coastline. Once these steps are complete, we proceed with composite activity definition generation, i.e., we provide a natural language description of an activity and ask the LLM to express it in the language of RTEC (prompt G). In what follows, we present prompts F*, F, E, T, and G. Prompt R is based on Definitions 2.2 and 2.4.

3.1 Simple and Statically Determined FVPs

In this step the goal is to demonstrate to the LLM the possible ways in which a composite activity may be expressed, i.e., as a simple or a statically determined FVP. To achieve this, we explain the differences between these types of FVP — see prompt F below. One should replace lines 8, 11, and 14 in prompt F with rules (1), (2), and (3) respectively. Moreover, one should replace line 16 with a simple FVP definition other than "within area" (see lines 5–14), and line 30 with a statically determined FVP definition other than "under way" (see lines 20–28). In the case of the chain-of-thought prompting, we provide two concrete example formalisations for each type of FVP — see lines 5–16 and 20–30 in prompt F. In contrast, in the case of few-shot prompting, i.e., prompt F without lines 7, 10, 13, and 22, we provide the natural language description of each composite activity and the corresponding formalisation, without presenting an explanation of the formalisation.

```
1 There are two ways in which a composite activity may be
  defined in the language of RTEC. In the first case, a
  composite activity definition may be specified by means
  of rules with initiatedAt(F=V,T) or terminatedAt(F=V,T)
  in their head. This is called a simple fluent definition.
2
3 The first body literal of an initiatedAt(F=V,T) rule is
  a positive happensAt predicate; this is followed by a
  possibly empty set of positive/negative happensAt and
  holdsAt predicates. Negative predicates are prefixed with
  'not' which expresses negation-by-failure. Below you may
  find two examples of composite activity definitions
  expressed as simple fluents.
4
5 Example 1: Given a composite maritime activity
  description, provide the rules in the language of RTEC.
  Composite Maritime Activity Description: 'withinArea'.
  This activity starts when a vessel enters an area of
  interest. The activity ends when the vessel leaves the
  area that it had entered. When there is a gap in signal
  transmissions, we can no longer assume that the vessel
  remains in the same area.
6
7 Answer: The activity 'withinArea' is expressed as a
  simple fluent. This activity starts when a vessel enters
  an area of interest. We use an 'initiatedAt' rule to
  express this initiation condition. The output is a
  boolean fluent named 'withinArea' with two arguments,
  i.e.,'Vessel' and 'AreaType'. We use one input event
  named 'entersArea' with two arguments 'Vessel' and
  'Area' and one fluent named 'areaType' with two arguments
  'Area' and 'AreaType'. This rule in the language of RTEC
  is the following:
8 <Rule (1)>
9
```

```
The activity 'withinArea' ends when a vessel leaves the
area that it had entered. We use a 'terminatedAt' rule
to describe this termination condition. We use one input
event named 'leavesArea' with two arguments 'Vessel' and
'Area' and one fluent named 'areaType' with two arguments
'Area' and 'AreaType'. This rule in the language of RTEC
language is:
<Rule (2)>

The activity 'withinArea' ends when a communication gap
starts. We use a 'terminatedAt' rule to express this
termination condition. We use one input event named
'gap_start' with one argument 'Vessel'. This rule in the
language of RTEC is:
<Rule (3)>

Example 2: <Definition 2>

A composite activity definition may be specified by
means of one rule with holdsFor(F=V, I) in its head. The
body of such a rule may include holdsFor(F'=V', I)
conditions, where F'=V' is different from F=V, as well
as the interval manipulation constructs of RTEC, i.e.,
union_all, intersect_all, and relative_complement_all. A
rule with holdsFor(F=V, I) in the head is called a
statically determined fluent definition. Below you may
find two examples of composite maritime activities
expressed as statically determined fluents.

Example 1: Given a composite maritime activity
description, provide the rules in the language of RTEC.
Composite Maritime Activity Description: 'underWay'.
This activity lasts as long as a vessel is not stopped.

Answer: The activity 'underWay' is expressed as a
statically determined fluent. Rules with 'holdsFor' in
the head specify the conditions in which a fluent holds.
We use a 'holdsFor' rule to describe that the 'underWay'
activity lasts as long as a vessel is not stopped. The
output is boolean fluent named 'underWay' with one
argument, i.e. 'Vessel'. We specify 'underWay' with the
use of the fluent 'movingSpeed'. More precisely, we
express 'underWay' as the disjunction of the three
values of 'movingSpeed', i.e. 'below', 'normal' and
'above'. Disjunction in 'holdsFor' rules is expressed by
means of 'union_all'. This rule is expressed in the
language of RTEC as follows:

  holdsFor(underWay(Vessel)=true, I) :-
     holdsFor(movingSpeed(Vessel)=below, I1),
     holdsFor(movingSpeed(Vessel)=normal, I2),
     holdsFor(movingSpeed(Vessel)=above, I3),
     union_all([I1,I2,I3], I).

Example 2: <Definition 2>
```

Prompt F

3.2 Events, FVPs, and Thresholds

In this step, we present the items of the input stream, i.e., the events and FVPs upon which RTEC reasons in order to detect, at run-time, the composite activities of interest. Moreover, we present the "background knowledge" of the application, which, in the maritime domain, includes a set of threshold values concerning the movement of vessels. The input events and FVPs, and the threshold values, may appear in the bodies of the rules expressing a composite activity. Prompts E and T present small fragments of the lists of input events and FVPs, and threshold values, for composite maritime activity recognition. By replacing line 6 with the remaining predicates and meanings, we complete the process of teaching the LLM the input events and FVPs, and threshold values.

```
1 You may use the following input events:
2
3 Input Event 1: change_in_speed_start(Vessel)
4 Meaning: 'Vessel' started changing its speed.
5
6 Input Event 2: <Predicate>
```

Prompt E

```
1 You may use a predicate named 'thresholds' with two
  arguments. The first argument refers to the threshold
  type and the second one to the threshold value.
  Threshold values can be used to perform mathematical
  operations and comparisons.
2
3 Threshold 1:thresholds(hcNearCoastMax,HcNearCoastMax)
4 Meaning: The maximum sailing speed that is safe for a
  vessel to have in a coastal area.
5
6 Threshold 2: <Predicate>
```

Prompt T

3.3 Rule Generation

Once the previous steps are complete, we proceed with rule generation, i.e., we provide a natural language description of each composite activity of interest and prompt the LLM to express it in the language of RTEC — prompt G below provides one such example. We explicitly guide the LLM to utilise the input events and FVPs, and threshold values that we have provided. Additionally, we instruct the LLM to take into consideration any of the activities that has been formalised so far. This way, we may construct a hierarchical knowledge base of activity definitions, according to which an activity definition depends on other "lower-level" definitions. Such hierarchical knowledge bases typically lead to succinct RTEC event descriptions, and pave the way for caching intermediate computations, and thus optimising run-time performance [4]. To generate the formalisations of all maritime activities of interest, one needs to replace line 5 in prompt G with the corresponding natural language descriptions.

```
1 Given a composite maritime activity description, provide
  the rules in RTEC formalization. You may use any of the
  aforementioned input events and fluents, and threshold
  values thresholds. You may use any of the output fluents
  that you have already learned.
2
3 Maritime Composite Activity Description - Communication
  gap: A communication gap starts when we stop receiving
  messages from a vessel. We would like to distinguish the
  cases where a communication gap starts (i) near some
  port and (ii) far from all ports. A communication gap
  ends when we resume receiving messages from a vessel.
4
5 <Maritime Composite Activitiy Description>
```

Prompt G

4 SIMILARITY METRIC

Unfortunately, LLM-generated activity definitions cannot always be supplied directly to RTEC, as they often include errors, in the form of, e.g., syntactic mistakes. To address this issue, we propose a similarity metric that compares an RTEC event description generated by an LLM, with a hand-crafted event description acting as the gold standard. Our metric supports human error correction, estimating the effort of manually correcting an LLM-generated activity definition, i.e. transforming the definition in a way that it may be used by RTEC. In the presence of a 'gold standard event description', the similarity metric allows us to perform a quantitative analysis of the LLM-generated definitions. This is in contrast to related approaches in the literature, where only qualitative assessments are performed.

Our metric extends the one proposed in [28], which computes the similarity between sets of ground expressions, i.e., atoms and terms that do not include variables. First, we outline the similarity metric of [28]. Subsequently, we extend this metric in order to make it suitable for comparing RTEC event descriptions.

4.1 Comparing Sets of Ground Expressions

We outline a distance function for ground expressions, after [31], and a distance function for sets of ground expressions, after [28].

Definition 4.1 (Distance between Ground Expressions). The distance between ground expressions $p(s_1, \dots, s_k)$ and $q(t_1, \dots, t_r)$ is defined as follows:

$$d^{\doteq}(p(s_1, \dots, s_k), q(t_1, \dots, t_r)) = \begin{cases} 0 & \text{if } k=r=0 \wedge p=q \\ \frac{1}{2k}\sum_{i=1}^{k} d^{\doteq}(s_i, t_i) & \text{if } k=r\neq 0 \wedge p=q \\ 1 & \text{if } k\neq r \vee p\neq q \end{cases} \quad (5)$$

■

Example 4.2 (Distance between Ground Expressions). Suppose that e_1 denotes happensAt(*entersArea*(*v42*, *a1*), *23*) and that e_2 denotes happensAt(*inArea*(*v42*, *a1*), *23*). Both e_1 and e_2 express that vessel '*v42*' enters the maritime area with ID '*a1*' at time-point *23*, but differ on the event name employed to denote this activity, i.e., e_1 uses *enterArea*, while e_2 uses *inArea*. The distance between e_1 and e_2 is calculated as follows:

$$d^{\doteq}(e_1,e_2) = \frac{1}{4}(d^{\doteq}(entersArea(v42, a1), inArea(v42, a1)) + d^{\doteq}(23,23))$$
$$= \frac{1}{4}(1+0) = 0.25$$

To compute $d^{\doteq}(e_1, e_2)$, $d^{\doteq}(entersArea(v42, a1), inArea(v42, a1))$ and $d^{\doteq}(23, 23)$, we follow, respectively, the second, the third and the first branch of equation (5). ◊

In order to compute the distance between two sets of ground expressions $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$, we need a cost matrix $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$, containing the pairwise distances between the expressions in $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$.

Definition 4.3 (Cost Matrix for Sets of Ground Expressions). Consider two sets of ground expressions: $\mathcal{E}^{\doteq}_a$, containing $e_{a,1}, \dots, e_{a,M}$, and $\mathcal{E}^{\doteq}_b$, containing $e_{b,1}, \dots, e_{b,K}$, where $M \geq K$. The cost matrix $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$ of $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$ is a square $M \times M$ matrix. The element in the i-th row and the j-th column of $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$ is:

$$\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{i,j} = \begin{cases} d^{\doteq}(e_{a,i}, e_{b,j}) & \text{if } j \leq K \\ 0 & \text{if } j > K \end{cases}$$

In the cases where $j > K$, the zero values capture the notion of unmatched expressions. ■

Example 4.4 (Cost Matrix for Sets of Ground Expressions). Consider set $\mathcal{E}^{\doteq}_a$, containing happensAt(*entersArea*(*v42*, *a1*), *23*), *areaType*(*a1*, *fishing*) and holdsAt(*underway*(*v42*) = true, *23*), denoted by $e_{a,1}$, $e_{a,2}$ and $e_{a,3}$, and set $\mathcal{E}^{\doteq}_b$, with expressions *areaType*(*a1*, *fishing*) and happensAt(*inArea*(*v42*, *a1*), *23*), denoted by $e_{b,1}$ and $e_{b,2}$. Based on Definition 4.3, the cost matrix of $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$ is:

$$\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b} = \begin{pmatrix} 1 & 0.25 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

$\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{1,2}$, e.g., contains the distance between ground expressions $e_{a,1}$ and $e_{b,2}$, which is equal to *0.25* (Example 4.2). Moreover, $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{1,1}$, i.e., the distance between $e_{a,1}$ and $e_{b,1}$, is *1*, because $e_{a,1}$ and $e_{b,1}$ concern different predicates (see Definition 4.1), while we set $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{1,3} = 0$ in order to be able to express that $e_{a,1}$ is left unmatched. ◊

Cost matrix $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$ may be used to deduce a mapping g between the expressions in sets $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$ that is optimal, in the sense that it minimises the sum of the distances between the paired expressions. A naive approach for finding g is to calculate the sum of the distances between paired expressions for each possible mapping and then select the mapping that yields the minimum such sum. This approach is too expensive; given that $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$ contain n expressions each, there are $n!$ possible mappings to consider. For this reason, we employ the Kuhn-Munkres algorithm, which derives the optimal mapping g with a worst-case cost of $O(n^3)$ [25]. Having computed g, the distance between $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$ is derived based on the distances of the expressions that are paired with mapping g.

Definition 4.5 (Distance between Sets of Ground Expressions). Consider two sets of ground expressions $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$, with sizes M and K, where $M \geq K$, the cost matrix $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$ of $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$, and the optimal mapping g of the ground expressions in $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$. The distance between $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$ is:

$$d^{\mathcal{E}^{\doteq}}(\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b) = \frac{1}{M}\left((M-K) + \sum_{(i,j)\in g} \mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{i,j}\right)$$

The term $M-K$ is used to penalise every unmatched ground expression by the greatest possible distance, i.e., *1*. ■

The similarity between two sets of ground expressions with distance d is $1-d$.

Example 4.6 (Distance between Sets of Ground Expressions). Consider the sets of ground expressions $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$, with cost matrix $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$, of Example 4.4. Using the Kuhn-Munkres algorithm, we derive the optimal mapping g between the expressions in $\mathcal{E}^{\doteq}_a$ and $\mathcal{E}^{\doteq}_b$, containing the following pairs: (*1*, *2*), (*2*, *1*), (*3*, *3*). Pair (*1*, *2*), e.g., matches expressions $e_{a,1}$ and $e_{b,2}$, while pair (*3*, *3*) indicates that expression $e_{a,3}$ of $\mathcal{E}^{\doteq}_a$ is left unmatched. g yields the minimum sum of distances that may be induced based on matrix $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}$, which is equal to $\mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{1,2} + \mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{2,1} + \mathbf{C}^{\mathcal{E}^{\doteq}_a, \mathcal{E}^{\doteq}_b}_{3,3} = 0.25$. Based on mapping g and Definition 4.5, the distance between $\mathcal{E}^{\doteq}_a$

and $\mathcal{E}_b^{\div}$ is:

$$d^{\mathcal{E}^{\div}}(\mathcal{E}_a^{\div}, \mathcal{E}_b^{\div}) = \frac{1}{3}\left((3-2) + \mathbf{C}_{1,2}^{\mathcal{E}_a^{\div},\mathcal{E}_b^{\div}} + \mathbf{C}_{2,1}^{\mathcal{E}_a^{\div},\mathcal{E}_b^{\div}} + \mathbf{C}_{3,3}^{\mathcal{E}_a^{\div},\mathcal{E}_b^{\div}}\right)$$
$$= \frac{1}{3}(1 + 0.25 + 0 + 0) = 0.4167$$

Thus, the similarity between $\mathcal{E}_a^{\div}$ and $\mathcal{E}_b^{\div}$ is $1-0.4167 = 0.5833$. ◊

4.2 Comparing Event Descriptions

We extend the distance metric in Definition 4.5 for comparing event descriptions in RTEC. Event descriptions are sets of rules defining FVPs, each containing a head atom and a set of body atoms. Definition 4.5 cannot be used to compare event descriptions because:

- Definition 4.5 concerns only ground expressions, while the atoms appearing in a rule may contain variables.
- Rules cannot be treated as sets of atoms; the head of a rule is not comparable with a body atom of a different rule.
- An event description is a set of rules, not a set of atoms.

First, we extend Definition 4.1 in order to consider non-ground expressions. Subsequently, we define the distance between two rules and the distance between two event descriptions.

Variables appearing in different rules may represent different concepts, even though they have the same name. Moreover, variables with different names may be referring to the same concept in different rules. We will define a distance function that assigns distance 0 to pairs of variables that are referring to the same concept, and distance 1 to every other pair of variables. We identify the concept to which a variable is referring in some rule by inspecting the instances of the variable in the rule. To specify these instances, we first define the tree representation of an expression and an instance of a variable in an expression based on its tree representation. Subsequently, we define the set of instances of a variable in a rule.

Definition 4.7 (Tree Representation of Expression). Consider an expression $p(s_1, \dots, s_k)$. If $k = 0$, i.e., p is a constant or a variable, then its tree representation is a single node with label p. Otherwise, the tree representation of $p(s_1, \dots, s_k)$ has a root node with label p and k children. For each $1 \le i \le k$, the i-th child of the root is the tree representation of expression s_i. ■

Example 4.8 (Tree Representation of Expression). Consider the tree representation t_h of $\mathsf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID}), T)$. The root of t_h has label happensAt and two children. Its first child is the tree representation of $\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID})$, i.e., a tree whose root has label *enterArea* and two children: the single nodes with labels *Vl* and *AreaID*. The second child of t_h is a single node with label T. ◊

Given the tree representation t of an expression $p(s_1, \dots, s_k)$, we use $t[(p, i)]$, where $1 \le i \le k$, to denote the sub-tree that is the i-th child of the root of t. We use $t[(p_1, i_1), (p_2, i_2), \dots (p_n, i_n)]$ as a shorthand for $t[(p_1, i_1)][(p_2, i_2)] \dots [(p_n, i_n)]$.

Definition 4.9 (Instance of Variable in Expression). Consider an expression u with tree representation t and a variable V that appears in u. If the sub-tree $t[(p_1, i_1), (p_2, i_2), \dots, (p_n, i_n)]$ of t is a single node with label V, then $[(p_1, i_1), (p_2, i_2), \dots, (p_n, i_n)]$ is an instance of variable V in expression u. ■

The instances of a variable V in an expression u may be derived with a depth-first search on the tree representation of u. We use $vi_r(V)$ to denote the list containing the instances of a variable V in the expressions included in rule r.

Example 4.10 (Instances of Variable in Rule). Consider rule (1), expressing the initiation of $\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true}$. The first instance of variable *Vl* in rule (1) is found in the head of the rule. There, *Vl* is the first argument of $\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType})$, which is, in turn, the first argument of expression '$=(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}), \mathsf{true})$', following a prefix notation for '='. The latter expression is the first argument of the head $\mathsf{initiatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true}, T)$ of rule (1). Moreover, there is an instance of variable *Vl* in condition $\mathsf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID}), T)$ of rule (1). There, *Vl* is the first argument of $\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID})$, which is the first argument of $\mathsf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID}), T)$. Therefore, list $vi_{r(1)}(\mathit{Vl})$ contains $[(\mathsf{initiatedAt}, 1), (=, 1), (\mathit{withinArea}, 1)]$ and $[(\mathsf{happensAt}, 1), (\mathit{entersArea}, 1)]$.

The remaining variables of rule (1) are *AreaType* and *AreaID*. $vi_{r(1)}(\mathit{AreaType})$ contains $[(\mathsf{initiatedAt}, 1), (=, 1), (\mathit{withinArea}, 2)]$ and $[(\mathit{areaType}, 2)]$, while $vi_{r(1)}(\mathit{AreaID})$ contains $[(\mathit{areaType}, 1)]$ and $[(\mathsf{happensAt}, 1), (\mathit{entersArea}, 2)]$. ◊

Definition 4.11 (Distance between Expressions). Suppose that rule r_1 (resp. r_2) contains the possibly non-ground expression $p(s_1, \dots, s_k)$ ($q(t_1, \dots, t_r)$). vi_{r_1} (vi_{r_2}) maps each variable in r_1 (r_2) to its list of instances in r_1 (r_2). The distance between expressions u_1 and u_2 based on vi_{r_1} and vi_{r_2} is defined as follows:

$$d(p(s_1, \dots, s_k), q(t_1, \dots, t_r), vi_{r_1}, vi_{r_2}) =$$
$$\begin{cases} 0 & \text{if } const(u_1) \wedge const(u_2) \wedge u_1 = u_2 \\ 0 & \text{if } var(u_1) \wedge var(u_2) \wedge vi_{r_1}(u_1) = vi_{r_2}(u_2) \\ 1 & \text{if } var(u_1) \wedge var(u_2) \wedge vi_{r_1}(u_1) \neq vi_{r_2}(u_2) \\ \frac{1}{2k}\sum\limits_{i=1}^{k} d(s_i, t_i, vi_{r_1}, vi_{r_2}) & \text{if } k = r \neq 0 \wedge p = q \\ 1 & \text{if } k \neq r \vee p \neq q \end{cases}$$

$const(x)$ (resp. $var(x)$) denotes that x is a constant (variable), while two lists of variable instances are considered equal iff they contain the same variable instances. ■

Definition 4.12 (Distance between Rules). Consider rules r_1 and r_2, with heads h_1 and h_2, and bodies b_1 and b_2, respectively. b_1 and b_2 contain, respectively, M and K conditions, where $M \ge K$. The cost matrix $\mathbf{C}^{b_1,b_2}$, containing the pairwise distances between the expressions in b_1 and b_2, is derived by Definition 4.3 using distance function d (see Definition 4.11) instead of $d^{\div}$, while g is the mapping between the expressions in b_1 and b_2 that minimises the sum of the distances between paired expressions with respect to $\mathbf{C}^{b_1,b_2}$. The distance $d^r(r_1, r_2)$ between rules r_1 and r_2 is defined as follows:

$$d^r(r_1, r_2) = \frac{1}{M+1}\left(d(h_1, h_2, vi_{r_1}, vi_{r_2}) + (M-K) + \sum_{\substack{(i,j)\in \\ g(\mathbf{C}^{b_1,b_2})}} \mathbf{C}_{i,j}^{b_1,b_2}\right)$$

■

The head h_1 (resp. h_2) of rule r_1 (r_2) is not comparable with any one of the conditions in body b_2 (b_1) of rule r_2 (r_1). Thus, Definition 4.12 compares the heads of rules r_1 and r_2 to each other, without considering them in mapping g.

Example 4.13 (Distance between Rules). Consider the following rules, expressing initiations of $\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true}$.

$$\begin{aligned} &\mathsf{initiatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{Area}), T), \\ &\quad \mathit{areaType}(\mathit{Area}, \mathit{AreaType}). \end{aligned} \tag{6}$$

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{withinArea}(Vl, AreaType) = \mathsf{true}, T) \leftarrow\\ &\quad \mathsf{happensAt}(\mathit{entersArea}(Vl, AreaID), T),\\ &\quad \mathit{areaType}(AreaType, AreaID).\end{aligned} \tag{7}$$

Rule (6) is generated from rule (1) by renaming variable *AreaID* as *Area*. Variable renaming does not affect the meaning of a rule. Thus, rules (1) and (6) are equivalent and their distance should be *0*. Rule (7) is the same as rule (1), with the exception that the arguments of *areaType* are given in reverse order. The order of the arguments matters when interpreting a predicate. Thus, rules (1) and (7) are not equivalent and their distance should be greater than *0*.

The list of instances of variable *Area* in rule (6) is the same as $vi_{r(1)}(AreaID)$ (see Example 4.10). Thus, according to Definition 4.11, the distance between $\mathsf{happensAt}(\mathit{entersArea}(Vl, AreaID), T)$ and $\mathsf{happensAt}(\mathit{entersArea}(Vl, Area), T)$, and the distance between $\mathit{areaType}(AreaID, AreaType)$ and $\mathit{areaType}(Area, AreaType)$, based on $vi_{r(1)}(AreaID)$ and $vi_{r(6)}(Area)$, are *0*. Therefore, following Definition 4.12, the distance between rule (1) and rule (6) is *0*.

Regarding the distance between rule (1) and rule (7), these rules differ only in the second condition of their bodies. Variables *AreaType* and *AreaID* have the following lists of instances in rule (7):

$$\begin{aligned}vi_{r(7)}(AreaType) =& [[(\mathsf{initiatedAt}, 1), (=, 1), (\mathit{withinArea}, 2)],\\ & [(\mathit{areaType}, 1)]]\\ vi_{r(7)}(AreaID) =& [[(\mathsf{happensAt}, 1), (\mathit{entersArea}, 2)],\\ & [(\mathit{areaType}, 2)]]\end{aligned}$$

Based on Example 4.10, $vi_{r(1)}(AreaType) \neq vi_{r(7)}(AreaType)$ and $vi_{r(1)}(AreaID) \neq vi_{r(7)}(AreaID)$. Therefore, according to Definition 4.11, it holds that $d(AreaType, AreaType, vi_{r(1)}, vi_{r(7)}) = 1$ and $d(AreaID, AreaID, vi_{r(1)}, vi_{r(7)}) = 1$. As a result, the distance between expressions containing variables *AreaType* and *AreaID* in rules (1) and (7) is greater than *0*. For instance, the distance between expression $\mathit{areaType}(AreaID, AreaType)$ of rule (1) and expression $\mathit{areaType}(AreaType, AreaID)$ of rule (7) is *0.5* (see Definition 4.11). Following Definition 4.12, the distance between rules (1) and (7) is:

$$d^r(r(1), r(7)) = \frac{1}{3}(0.015625+0+0.0625+0.5) = 0.1667 \qquad \diamond$$

Definition 4.14 (Event Description Distance). Consider two event descriptions in RTEC: KB_1, containing M rules, and KB_2, containing K rules, where $M \geq K$. The cost matrix $\mathbf{C}^{KB_1,KB_2}$, containing the pairwise distances between the rules in KB_1 and KB_2, is derived by Definition 4.3 using distance function d^r (see Definition 4.12) instead of $d^{\div}$, while g is the mapping between the rules in KB_1 and KB_2 that minimises the sum of the distances between paired rules with respect to $\mathbf{C}^{KB_1,KB_2}$. The distance $D(KB_1, KB_2)$ between event descriptions KB_1 and KB_2 is defined as follows:

$$D(KB_1, KB_2) = \frac{1}{M}\left((M-K) + \sum_{(i,j)\in g(\mathbf{C}^{KB_1,KB_2})} \mathbf{C}^{KB_1,KB_2}_{i,j}\right) \quad \blacksquare$$

The similarity between two event descriptions with distance d is $1-d$.

5 EMPIRICAL ANALYSIS

5.1 Experimental Setup

We evaluated our approach on composite activity recognition for maritime situational awareness. As input, we consider Automatic Identification System (AIS) position signals emitted by vessels, containing information about their heading, speed and navigational status [5]. We employed a real, publicly available dataset[2] containing 18M AIS position signals emitted from 5K vessels sailing in the Atlantic Ocean around the port of Brest, France, between October 2015–March 2016. The task was to generate an event description in RTEC that includes accurate definitions for composite maritime activities, such as trawling and ship-to-ship transfer. To generate the maritime activity definitions, we employed GPT-4, GPT-4o, o1[3], Gemma-2[4], Mistral[5] and LLaMa-3[6], using the OpenAI API and Groq API.

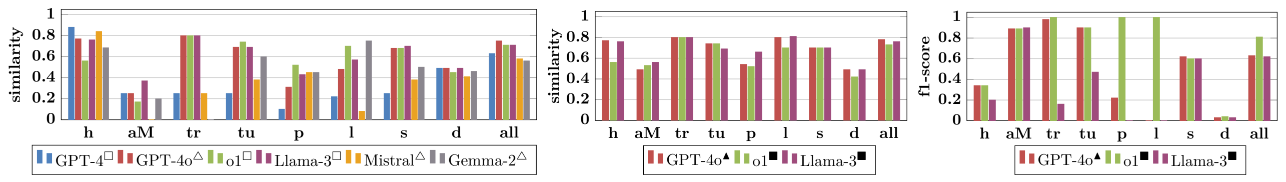
5.2 Experimental Results

Syntactic Similarity. In the first set of experiments, we applied our prompting method on GPT-4, GPT-4o, o1, Llama-3, Mistral and Gemma-2, in order to assess the quality of the event descriptions they generate. We used two types of prompting for each LLM, i.e., few-shot and chain-of-thought prompting (see Section 3). We use $X^{\square}$ (resp. $X^{\triangle}$) to denote the event description generated by LLM X via few-shot (chain-of-thought) prompting. As the gold standard for our comparison, we employed an event description containing composite maritime activity definitions that were hand-crafted with the help of domain expects. This event description is described in [33] and is publicly available[1].

For each LLM X, we calculated the similarities of event descriptions $X^{\square}$ and $X^{\triangle}$ with our hand-crafted event description, according to Definition 4.14, and identified the one, i.e., $X^{\square}$ or $X^{\triangle}$, that yields the highest similarity value. Figure 2a presents these similarity values, along with the prompting technique that enabled each LLM to generate them. The first 8 bar-groups express similarity values for the definitions of 8 composite maritime activities of interest, while the last bar-group denotes the average similarity value for all activity definitions in our event descriptions. Note that employing chain-of-thought prompting does not necessarily lead to more accurate definitions; long chains-of-thought, e.g., may lead to a decrease in accuracy [39]. In the case of 'trawling' (see the bar-group labeled 'tr'), e.g., GPT-4o$^{\triangle}$, o1$^{\square}$ and Llama-3$^{\square}$ generated definitions that yield a high similarity value, as they were able express most of the conditions in our hand-crafted definition of trawling, while introducing only one redundant condition. In contrast, the corresponding definitions generated by GPT-4$^{\square}$ and Mistral$^{\triangle}$ did not match any one of the conditions in our hand-crafted definition, yielding a much lower similarity. Gemma-2$^{\triangle}$ expressed 'trawling' as a simple fluent, while the hand-crafted rules express it as a statically determined fluent, resulting in a similarity of 0.

Qualitative Error Assessment. LLM-generated event descriptions may include errors, in the form of, e.g., syntactic mistakes or conditions with undefined activities, which typically lead to a drop in their similarity with respect to the gold standard. The errors found in the event descriptions generated in our evaluation typically fall within one of the following categories. The first category of errors includes minor divergences between generated event descriptions and the gold standard in the names chosen for expressions denoting events, composite activities and background knowledge. The second category concerns modeling a composite activity definition using a different

[2]https://zenodo.org/record/1167595
[3]https://chatgpt.com/
[4]https://huggingface.co/blog/gemma2
[5]https://chat.mistral.ai/chat
[6]https://www.llama.com/



(a) Similarity values of LLM-generated definitions. **(b) Similarities after syntactic changes.** **(c) Predictive accuracy.**

Figure 2: 'h', 'aM', 'tr', 'tu', 'p', 'l', 's' and 'd' stand for 'high speed near coast', 'anchored or moored', 'trawling', 'tugging' 'pilot boarding', 'loitering', 'search-and-rescue' and 'drifting'. For each LLM, we report only the prompting scheme, i.e., few-shot or chain-of-thought, with the highest similarity.

type of fluent than the one used in the hand-crafted event description. For instance, GPT-4o uses a statically determined fluent to specify 'movingSpeed', which is defined with a simple fluent in the hand-crafted rules. The third category concerns generated definitions that cannot be used in practice, because their conditions include composite activities that are not defined in the generated event description. Lastly, we observed that LLMs often fail at capturing definitions that include multiple operations between activities, such as disjunction, conjunction and negation. For instance, GPT4o generated a definition of 'loitering', which is incorrect, as it uses 'intersect_all' in the place of 'union_all'.

Performance on CER. In the second set of experiments, our goal was to assess the predictive accuracy of LLM-generated event descriptions, when used for detecting composite maritime activities. We evaluated the three event descriptions with the highest similarity values, i.e., GPT-4o$^{\triangle}$, o1$^{\square}$ and Llama-3$^{\square}$ (see Figure 2a). Unfortunately, these event descriptions cannot be used directly by RTEC, as they include minor syntactic errors, such as incorrect names for constants and predicates. In event description o1$^{\square}$, e.g., we had to rename constant 'trawlingArea' as 'fishing', because the predicates expressing background knowledge, such as *areaType* (see, e.g., rule (1)), use the latter constant to denote fishing areas. After performing the minimum required changes on event descriptions GPT-4o$^{\triangle}$, o1$^{\square}$ and Llama-3$^{\square}$, in order to be compatible with RTEC, which led to event descriptions GPT-4o$^{\blacktriangle}$,o1$^{\blacksquare}$ and Llama-3$^{\blacksquare}$, we measured their similarity with our hand-crafted event description. Figure 2b outlines the corresponding similarity values, demonstrating that our changes were minor, i.e., led to a small increase in the average similarity score of a generated definition, compared to the original score (see Figure 2a).

We instructed RTEC to detect composite maritime activities over real streams of vessel movement, included in the dataset of Brest, based on the maritime activity definitions in event descriptions GPT-4o$^{\blacktriangle}$, o1$^{\blacksquare}$ and Llama-3$^{\blacksquare}$. We collected the maritime activity instances that were detected using each event description, and compared them with the ones detected using our hand-crafted event description. For each maritime activity, we calculated the time-points (expressing seconds) at which the activity was recognised based on both its LLM-generated and its hand-crafted definition, making up the 'true positives'. Similarly, we derived the 'false positives' (resp. 'false negatives') by computing the time-points at which the activity was detected based on its LLM-generated (hand-crafted) definition, but not based on its hand-crafted (LLM-generated) definition. Based on these metrics, we computed the predictive accuracy of event descriptions GPT-4o$^{\blacktriangle}$, o1$^{\blacksquare}$ and Llama-3$^{\blacksquare}$ for each composite maritime activity. Figure 2c presents our results, which demonstrate that event description o1$^{\blacksquare}$ leads to a higher predictive accuracy than event descriptions GPT-4o$^{\blacktriangle}$ and Llama-3$^{\blacksquare}$. While all three event descriptions contained comparably accurate definitions for most simple FVPs, we observed that o1$^{\blacksquare}$ included more accurate definitions for some statically determined FVPs, such as the ones expressing 'loitering' and 'pilot boarding' (see Figure 2a). The definition of loitering, e.g., stipulates that the vessel needs to be 'stopped' or 'sail at a low speed'. Though these two activities are mutually exclusive, GPT-4o$^{\blacktriangle}$ and Llama-3$^{\blacksquare}$ define loitering in terms of their conjunction, confusing union_all with intersect_all, and thus leading to a rule that is never satisfied. In contrast, o1$^{\blacksquare}$ defines loitering using the disjunction of 'stopped' or 'sail at a low speed', leading to a definition that, although not being syntacticly equivalent with the hand-crafted one, effectively expresses the same meaning, and thus results in a perfect f1-score.

6 SUMMARY AND FURTHER WORK

We proposed a method that employs pre-trained LLMs in order to generate RTEC specifications for composite activities based on natural language descriptions. Contrary to machine learning approaches [22, 29], our method does not require labels for composite activities. Moreover, in order to measure the human effort required to correct LLM-generated definitions, we proposed a similarity metric for RTEC specifications. Our experimental evaluation on the maritime domain demonstrated the effectiveness of our approach.

In principle, our approach may be used in other domains, such as composite activity recognition for vehicle fleet management [34]. Prompt R may be re-used as it is, while the prompts F, E, and T may be customised with domain-specific knowledge. Testing our method in other domains is direction for further work. Furthermore, we aim to take advantage of the recent advances in open foundational models, such as OLMo[7].

ACKNOWLEDGMENTS

This work was supported partly by the University of Piraeus Research Center and partly by the EU-funded CREXDATA project (grant agreement No 101092749).

REFERENCES

[1] Elias Alevizos, Alexander Artikis, and Georgios Paliouras. 2022. Complex event forecasting with prediction suffix trees. *VLDB J.* 31 (2022).

[2] Elias Alevizos, Alexander Artikis, and Georgios Paliouras. 2024. Complex Event Recognition with Symbolic Register Transducers. *Proc. VLDB Endow.* 17, 11 (2024), 3165–3177.

[3] Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. 2022. Modeling and Reasoning in Event Calculus using Goal-Directed Constraint Answer Set Programming. *Theory Pract. Log. Program.* 22, 1 (2022), 51–80.

[4] Alexander Artikis, Marek J. Sergot, and Georgios Paliouras. 2015. An Event Calculus for Event Recognition. *IEEE Trans. Knowl. Data Eng.* 27, 4 (2015), 895–908.

[7] https://allenai.org/olmo

[5] Alexander Artikis and Dimitris Zissis (Eds.). 2021. *Guide to Maritime Informatics*. Springer. https://doi.org/10.1007/978-3-030-61852-0

[6] Peter Baumgartner. 2021. Combining Event Calculus and Description Logic Reasoning via Logic Programming. In *FroCoS*. 98–117.

[7] Harald Beck, Minh Dao-Tran, and Thomas Eiter. 2018. LARS: A Logic-based framework for Analytic Reasoning over Streams. *Artif. Intell.* 261 (2018), 16–70.

[8] Harald Beck, Thomas Eiter, and Christian Folie. 2017. Ticker: A system for incremental ASP-based stream reasoning. *Theory Pract. Log. Program.* 17, 5-6 (2017), 744–763.

[9] Marco Bucchi, Alejandro Grez, Andrés Quintana, Cristian Riveros, and Stijn Vansummeren. 2022. CORE: a COmplex event Recognition Engine. *Proc. VLDB Endow.* 15, 9 (2022), 1951–1964.

[10] Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni. 2009. Commitment Tracking via the Reactive Event Calculus. In *IJCAI*. 91–96.

[11] Keith L. Clark. 1977. Negation as Failure. In *Logic and Data Bases*. Plemum Press, 293–322.

[12] Erica Coppolillo, Francesco Calimeri, Giuseppe Manco, Simona Perri, and Francesco Ricca. 2024. LLASP: Fine-tuning Large Language Models for Answer Set Programming. In *KR*. 834–844.

[13] Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. 2019. A Distributed Approach to LARS Stream Reasoning (System paper). *Theory Pract. Log. Program.* 19, 5-6 (2019), 974–989.

[14] Nicola Falcionelli, Paolo Sernani, Albert Brugués de la Torre, Dagmawi Neway Mekuria, Davide Calvaresi, Michael Schumacher, Aldo Franco Dragoni, and Stefano Bromuri. 2019. Indexing the Event Calculus: Towards practical human-readable Personal Health Systems. *Artif. Intell. Medicine* 96 (2019), 154–166.

[15] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.* 17, 5 (2024), 1132–1145.

[16] Lars George, Bruno Cadonna, and Matthias Weidlich. 2016. IL-Miner: Instance-Level Discovery of Complex Event Patterns. *Proc. VLDB Endow.* 10, 1 (2016), 25–36.

[17] Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. 2020. Complex event recognition in the Big Data era: a survey. *VLDB J.* 29, 1 (2020), 313–352.

[18] Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. 2021. A Formal Framework for Complex Event Recognition. *ACM Trans. Database Syst.* 46, 4 (2021), 16:1–16:49.

[19] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In *NeurIPS*.

[20] Adam Ishay, Zhun Yang, and Joohyung Lee. 2023. Leveraging Large Language Models to Generate Answer Set Programs. In *KR*. 374–383.

[21] Özgür Kafali, Alfonso E. Romero, and Kostas Stathis. 2017. Agent-oriented activity recognition in the event calculus: An application for diabetic patients. *Comput. Intell.* 33, 4 (2017), 899–925.

[22] Nikos Katzouris, Georgios Paliouras, and Alexander Artikis. 2023. Online Learning Probabilistic Event Calculus Theories in Answer Set Programming. *Theory Pract. Log. Program.* 23, 2 (2023), 362–386.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *NeurIPS*.

[24] R. Kowalski and M. Sergot. 1986. A Logic-Based Calculus of Events. *New Gen. Computing* 4, 1 (1986), 67–96.

[25] H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97.

[26] Periklis Mantenoglou, Dimitrios Kelesis, and Alexander Artikis. 2023. Complex Event Recognition with Allen Relations. In *KR*. 502–511.

[27] Periklis Mantenoglou, Manolis Pitsikalis, and Alexander Artikis. 2022. Stream Reasoning with Cycles. In *KR*. 544–553.

[28] Evangelos Michelioudakis, Alexander Artikis, and Georgios Paliouras. 2019. Semi-supervised online structure learning for composite event recognition. *Mach. Learn.* 108, 7 (2019), 1085–1110.

[29] Evangelos Michelioudakis, Alexander Artikis, and Georgios Paliouras. 2024. Online semi-supervised learning of composite event rules by combining structure and mass-based predicate similarity. *Mach. Learn.* 113, 3 (2024), 1445–1481.

[30] Marco Montali, Fabrizio Maria Maggi, Federico Chesani, Paola Mello, and Wil M. P. van der Aalst. 2013. Monitoring business constraints with the event calculus. *ACM Trans. Intell. Syst. Technol.* 5, 1 (2013), 17:1–17:30.

[31] Shan-Hwei Nienhuys-Cheng. 1997. Distance Between Herbrand Interpretations: A Measure for Approximations to a Target Concept. In *Inductive Logic Programming, 7th International Workshop, ILP-97, Prague, Czech Republic, September 17-20, 1997, Proceedings (Lecture Notes in Computer Science)*, Nada Lavrac and Saso Dzeroski (Eds.), Vol. 1297. Springer, 213–226.

[32] Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In *Findings of the Association of Computational Linguistics: EMNLP*. 3806–3824.

[33] Manolis Pitsikalis, Alexander Artikis, Richard Dreo, Cyril Ray, Elena Camossi, and Anne-Laure Jousselme. 2019. Composite Event Recognition for Maritime Monitoring. In *DEBS*. 163–174.

[34] Efthimis Tsilionis, Alexander Artikis, and Georgios Paliouras. 2022. Incremental Event Calculus for Run-Time Reasoning. *J. Artif. Intell. Res.* 73 (2022), 967–1023.

[35] Przemyslaw Andrzej Walega, Mark Kaminski, and Bernardo Cuenca Grau. 2019. Reasoning over Streaming Data in Metric Temporal Datalog. In *AAAI*. 3092–3099.

[36] Przemyslaw Andrzej Walega, Mark Kaminski, Dingmin Wang, and Bernardo Cuenca Grau. 2023. Stream reasoning with DatalogMTL. *J. Web Semant.* 76 (2023), 100776.

[37] Zhongsheng Wang, Jiamou Liu, Qiming Bao, Hongfei Rong, and Jingfeng Zhang. 2024. ChatLogic: Integrating Logic Programming with Large Language Models for Multi-Step Reasoning. In *IJCNN*. 1–8.

[38] Walker M. White, Mirek Riedewald, Johannes Gehrke, and Alan J. Demers. 2007. What is "next" in event processing?. In *PODS*. 263–272.

[39] Zhun Yang, Adam Ishay, and Joohyung Lee. 2023. Coupling Large Language Models with Logic Programming for Robust and General Reasoning from Text. In *Findings of the Association of Computational Linguistics: ACL*. 5186–5219.

[40] Bo Zhao, Han van der Aa, Thanh Tam Nguyen, Quoc Viet Hung Nguyen, and Matthias Weidlich. 2021. EIRES: Efficient Integration of Remote Data in Event Stream Processing. In *SIGMOD*. 2128–2141.