Observation Strategies for Event Detection with Incidence on Runtime Verification

Marco Alberti[1], Pierangelo Dell'Acqua[2], and Luís Moniz Pereira[1]

[1] Centro de Inteligência Artificial (CENTRIA), Departamento de Informática
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Caparica (Portugal)
m.alberti@fct.unl.pt,lmp@di.fct.unl.pt

[2] Dept. of Science and Technology, ITN
Linköping University
Norrköping (Sweden)
pierangelo.dellacqua@liu.se

Abstract

In many applications, it is required to detect the occurrence of an event in a system, which entails observing the system. Observation can be costly, so it makes sense to try and reduce the number of observations, without losing certainty about the event's occurrence. In this paper, we propose a formalization of the problem. We show (formally) that when the event to be detected follows a discrete spatial or temporal pattern, it is possible to reduce the number of observations. We provide an experimental evaluation of algorithms for this purpose. We apply the result to verification of linear temporal logics formulæ. Finally, we discuss possible generalizations, and how event detection and related applications can benefit from logic programming techniques.

1 Introduction

An event is an observable state in a system, that may occur at some point in space and time, and event detection is a crucial process in many applications (such as monitoring, runtime verification, diagnosis, intention recognition, and more [9]). In order to detect events, it is naturally necessary to observe the system where the event may occur.

However, observation may be a costly process. In this case, observing the system at all spatial or temporal points may be undesirable, and a question arises: is it possible to reduce the number of necessary observations, without losing certainty about the event's occurrence?

In this paper we formalize the problem, and show that, for the simple, but significant case where the observation space can be modelled as the set of natural numbers, the answer is yes.

The paper is structured as follows. In Section 2, we formulate the problem in a general setting. In Section 3, we study the problem for the special case where the temporal or spatial event structure can be described by a set of natural numbers. In Section 4, we compare experimentally a complete and an approximation algorithm. In Section 5, we give a formulation of the problem suitable for runtime verification of Linear Temporal Logic formulæ. In Section 6, we discuss possible extensions and generalizations of our approach, and the applications of techniques borrowed from logic programming to problems beyond simple event detection.

2 Event detection in general

2.1 Coordinate system

A coordinate system defines points where events can happen. Coordinates can be interpreted, for instance, as spatial, temporal, or both. We only require sum to be defined over coordinates.

Definition 1. *A **coordinate system** $\mathcal{S}$ is the cartesian product of N sets, closed under the sum operation.*

Each of the sets can be continuous (e.g., the set $\mathbb{R}$ of real numbers), or discrete (e.g., the sets $\mathbb{N}$ of natural numbers or $\mathbb{Z}$ of integer numbers).

2.2 Patterns

A pattern associates a value taken from a **value set** $\mathcal{V}$ to (some) points in the coordinate system.

In the simplest case, $\mathcal{V}$ will have only one element. This representation is satisfactory in cases where the value is not significant, and the relevant information is the set of points where the pattern is defined.

Definition 2. *Given a coordinate system $\mathcal{S}$, a **pattern** $\mathcal{P}$ over $\mathcal{S}$ is a partial function from $\mathcal{S}$ to some value set $\mathcal{V}$.*

As usual, we call the pattern's **domain** the set of points where the pattern is defined.

$\mathcal{S}$	$\mathcal{V}$	$\mathcal{P}$
$\mathbb{N}$	$\{1\}$	$\{\langle 0,1\rangle, \langle 4,1\rangle, \langle 5,1\rangle\}$
$\mathbb{R}^2$	$\mathbb{R}$	$\{\langle\langle 0,0\rangle, 0\rangle\}$
$\mathbb{R}^2$	$\mathbb{R}$	$\{\langle\langle x,y\rangle, x\rangle \mid x \in \mathbb{R} \wedge y \in \mathbb{R}\}$
$\mathbb{R}^2$	$\mathbb{R}$	$\{\langle\langle x,y\rangle, 1\rangle \mid 0 \leq x \leq 1 \wedge 0 \leq y \leq 1\}$

Table 1: Example patterns

An event is an association of values to points in a coordinate system that follows a given pattern. It is defined as the translation of its pattern by an offset.

Definition 3. *Given a pattern $\mathcal{P}$ and $\tau \in \mathcal{S}$, the* ***event*** $\mathcal{E}_{\mathcal{P}}(\tau)$ *of pattern $\mathcal{P}$ and offset τ is the set $\{\langle \tau + p, \mathcal{P}(p)\rangle \mid p \in \operatorname{dom} \mathcal{P}\}$.*

We call *unfolding* the set of all possible events of a given pattern.

Definition 4. *Given a pattern $\mathcal{P}$ in a coordinate system $\mathcal{S}$, the* ***unfolding*** *$\mathcal{U}_{\mathcal{P}}$ of $\mathcal{P}$ is the set $\{\mathcal{E}_{\mathcal{P}}(\tau) \mid \tau \in \mathcal{S}\}$.*

Definition 5. *An event e's* ***observability set*** *$\mathcal{O}(e)$ is e's projection over $\mathcal{S}$.*

Intuitively, the observability set is the set of points in the coordinate system where the event can be observed.

Example 1. *If $\mathcal{P}$ is the first pattern in Table 1, the event $\mathcal{E}_{\mathcal{P}}(5)$ is $\{\langle 5, 1\rangle, \langle 9, 1\rangle, \langle 10, 1\rangle\}$, and its observability set is $\{5, 9, 10\}$.*

A set of points in the coordinate system that is guaranteed to intersect the observability set of any event with a given pattern can be understood as the set of points to observe to decide about the occurrence of such an event. We say that such a set *covers* the pattern or is one of the pattern's *covering sets.* For example, if the coordinate system represents a spatial area, a covering set can be understood as a set of positions to deploy sensors (as in wireless sensor networks applications).

Definition 6. *A set $\mathcal{C} \subseteq \mathcal{S}$* ***covers*** *a pattern $\mathcal{P}$ if and only if for each $\tau \in \mathcal{S}$ there exists an element of $\mathcal{O}(\mathcal{E}_{\mathcal{P}}(\tau))$ that belongs to $\mathcal{C}$.*

Example 2. *Consider a pattern $\mathcal{P}$, whose domain is $\{0, 1, 2\}$. $\{3n \mid n \in \mathbb{N}\}$ covers $\mathcal{P}$. $\mathcal{C} \triangleq \{1, 2\}$ does not cover $\mathcal{P}$, because, for instance, 3 belongs to the observability set of the event of pattern $\mathcal{P}$ and offset 3 and does not belong to $\mathcal{C}$. For the same reason, $\{4n \mid n \in \mathbb{N}\}$ does not cover $\mathcal{P}$.*

Example 3. *$\mathcal{C} \triangleq \mathbb{Z}\times\mathbb{Z}$ covers the last pattern in Table 1. Indeed, the observability set of an event of offset $\tau \triangleq \langle \tau_x, \tau_y\rangle$ is*
$\{\langle x, y\rangle \mid \tau_x \leq x \leq \tau_x + 1 \ \wedge\ \tau_y \leq y \leq \tau_y + 1\}$. It contains $\langle \lfloor \tau_x \rfloor + 1, \lfloor \tau_y \rfloor + 1\rangle$ (an element of $\mathcal{C}$), because, for any $\alpha \in \mathbb{R}$, $\alpha \leq \lfloor \alpha \rfloor + 1 \leq \alpha + 1$.

2.3 The sampling problem

If the application requires to determine the occurrence of an event with a given pattern, it is of interest to determine a set of points in the coordinate system that guarantees that, if an event occurs, it will be observed.

Definition 7. *Given a pattern $\mathcal{P}$ over $\mathcal{S}$, the* ***sampling problem*** *consists of finding a set $\mathcal{C} \subseteq \mathcal{S}$ that covers $\mathcal{P}$.*

The general problem has a trivial solution ($\mathcal{S}$), but the solution may be required to enjoy some property. A reasonable requirement is for it to minimize a cost function. If the cost of observation is a constant, that translates to minimum cardinality. However, applications can impose more requirements. For instance, if the points in the coordinate system represent time points, then it may be required for the covering set to intersect observability sets at or near their minimum, in order to detect events as soon as possible.

3 Sampling problem for natural patterns

In this section, we consider the case where the coordinate system is the set $\mathbb{N}$ of natural numbers.

Definition 8. *A* ***natural pattern*** *is a pattern over the set* $\mathbb{N}$ *of natural numbers.*

A natural interpretation for this case is that the coordinate system represents a sequence of time points at which the system can be observed.

The following result allows for a reduction of the sampling problem for natural patterns. First we provide two definitions.

Definition 9. *Let* $\mathcal{I}$ *be a finite set of natural numbers, and let* $M \triangleq \max \mathcal{I}$.
For an integer k, *let* $\sigma_k \triangleq \{(p+k) \texttt{ mod } (M+1) \mid p \in \mathcal{I}\}$.
$\mathcal{I}$*'s* ***circular repetition*** *is the collection of sets*
$\mathcal{U} \triangleq \{\sigma_k \mid k \in [0 \mathrel{..} M]\}$

Definition 10. *Let* $\mathcal{P}$ *be a natural pattern.*
Then a set with non-empty intersection with each element of the circular repetition of $\operatorname{dom} \mathcal{P}$ *is a* ***covering shape*** *of* $\mathcal{P}$*'s.*

Theorem 1. *Let* $\mathcal{P}$ *be a natural pattern and let* $M \triangleq \max \operatorname{dom} \mathcal{P}$.
Let $\mathcal{S}_C$ *be a covering shape of* $\mathcal{P}$*'s.*
Then the set $\mathcal{C} \triangleq \{q + k(M+1) \mid q \in \mathcal{S}_C \wedge k \in \mathbb{N}\}$ *covers* $\mathcal{P}$.

Proof. Consider a generic event e of pattern $\mathcal{P}$ and offset τ. Let $R \triangleq \tau \texttt{ mod } (M+1)$. Consider σ_R. By hypothesis, it intersects $\mathcal{S}_C$ in (at least) a natural number q. Since $q \in \sigma_R$, there exists $p \in \operatorname{dom} \mathcal{P}$ such that $q = (p+R) \texttt{ mod } (M+1)$. Since $p \in \operatorname{dom} \mathcal{P}$, $\tau + p \in \mathcal{O}(e)$. Since τ is equal to R modulo $M+1$, $\tau+p$ is equal to $R+p$ modulo $M+1$, which is equal to q modulo $M+1$. Therefore, $\exists k \mid \tau + p = q + k(M+1)$. Therefore, also $\tau + p \in \mathcal{C}$.

Concluding, the observability set of a generic event of pattern $\mathcal{P}$ intersects $\mathcal{C}$; that is, $\mathcal{C}$ covers $\mathcal{P}$. □

Therefore, the sampling problem can be reduced to finding a set that intersects a collection of $M+1$ subsets of $[0 \mathrel{..} M]$. The periodic repetition of such set, with period $(M+1)$, covers the pattern. A trivial, and uninteresting, solution is $[0 .. M]$ itself (which corresponds to observing the system at all time points). However, finding such a set with minimum cardinality is a well known NP-hard problem: the *minimum hitting set* problem.

For applications, it seems reasonable to consider patterns whose domain contains 0. However, the following result lets us reduce to this case, for generic pattern domains.

Definition 11. *Given a natural pattern $\mathcal{P}$, let $m \triangleq \min \operatorname{dom} \mathcal{P}$. Then the **normalization** of $\mathcal{P}$ is the pattern $\overline{\mathcal{P}} \triangleq \{\langle p, \mathcal{P}(p+m)\rangle \mid p+m \in \operatorname{dom} \mathcal{P}\}$.*

Theorem 2. *Let $\mathcal{P}$ be a natural pattern. Then an event of pattern $\mathcal{P}$ is also an event of pattern $\overline{\mathcal{P}}$.*

Proof. $\mathcal{E}_{\mathcal{P}}(\tau) \triangleq \{\langle \tau+p, \mathcal{P}(p)\rangle \mid p \in \operatorname{dom} \mathcal{P}\}$ which, by substituting $q+m$ for p, is equal to $\{\langle \tau+q+m, \mathcal{P}(q+m)\rangle \mid q+m \in \operatorname{dom} \mathcal{P}\} = \mathcal{E}_{\overline{\mathcal{P}}}(\tau+m)$. □

Therefore, a set that covers the normalized pattern also covers the original pattern. This lets us solve the sampling problem for the normalized pattern, whose domain's maximum element is in general lesser, and reduce complexity and approximation ratio, if an approximation algorithm is used (see Section 4).

The following definition gives a measure of how much the number of necessary observation is reduced by observing the system only at a pattern's covering set, instead of at all points.

Definition 12. *Given a normalized pattern $\mathcal{P}$ and a covering shape $\mathcal{S}_C$ of $\mathcal{P}$'s, the corresponding **sampling ratio** is $\mathcal{R}_{\mathcal{S}_C,\mathcal{P}} \triangleq \frac{|\mathcal{S}_C|}{1+\max \operatorname{dom} \mathcal{P}}$.*

$\lvert\operatorname{dom} \mathcal{P}\rvert$	2	4	6	8	10	12	14	16	18	20
$\mathcal{R}_{\mathcal{S}_C,\mathcal{P}}$	0.52	0.32	0.24	0.19	0.15	0.14	0.12	0.1	0.1	0.1

Table 2: Sampling ratios.

Example 4. *Table 2 shows average sampling ratios computed on random patterns with domain maximum of 20, for different values of the domain cardinality.*

Example 5. *Let a pattern $\mathcal{P}$'s domain be $[0 .. M]$. $\operatorname{dom} \overline{\mathcal{P}}$'s circular repetition is $\mathcal{U} \triangleq \{\sigma_0, \ldots, \sigma_M\}$, where for $i \in [0 .. M]$ $\sigma_i = [0 .. M]$. A minimum hitting set for $\mathcal{U}$ is $\{i\}$, for any $i \in [0 .. M]$. For each i, a covering set is $\{i + k(M+1) \mid k \in \mathbb{N}\}$. The corresponding sampling ratio is $\frac{1}{M+1}$, confirming the intuitive idea that, if an event lasts $M+1$ points, it is sufficient to observe every $(M+1)$-th point to detect its occurrence.*

Example 6. *Consider a pattern $\mathcal{P}$ whose domain is* $\{8, 10, 13\}$. *Then* $\operatorname{dom} \overline{\mathcal{P}} = \{0, 2, 5\}$. $\operatorname{dom} \overline{\mathcal{P}}$*'s circular repetition is* $\mathcal{U} \triangleq \{\sigma_0, \ldots, \sigma_5\}$, *where* $\sigma_0 \triangleq \{0, 2, 5\}$, $\sigma_1 \triangleq \{0, 1, 3\}$, $\sigma_2 \triangleq \{1, 2, 4\}$, $\sigma_3 \triangleq \{2, 3, 5\}$, $\sigma_4 \triangleq \{0, 3, 4\}$, *and* $\sigma_5 \triangleq \{1, 4, 5\}$. $\overline{\mathcal{P}}$*'s covering shape* $\mathcal{S}_C$ *is required to have non-empty intersection with all the sets in* $\mathcal{U}$*; for instance, a possible covering shape is* $\mathcal{S}_C \triangleq \{0, 1, 3\}$. *It can be easily verified that its cardinality is minimal. The corresponding covering set of* $\mathcal{P}$*'s is* $\mathcal{C} \triangleq \{q + 6k \mid q \in \{0, 1, 3\} \wedge k \in \mathbb{N}\} = \{0, 1, 3, 6, 7, 9, 12, \ldots\}$, *which also covers* $\mathcal{P}$, *and the sampling ratio is* $\mathcal{R}_{\mathcal{S}_C, \overline{\mathcal{P}}} = 3/6 = 1/2$.

4 Algorithms and experimental evaluation

In this section, we discuss solution methods for the minimum hitting set problem that the sampling problem reduces to, for natural patterns, as shown in section 3. We describe a complete algorithm, and we discuss reduction to the set cover problem, and its solution by means of a well known approximation polynomial algorithm. We compare experimentally the two algorithms.

Complete algorithm The algorithm is based on a total order of all the elements of $2^{[0\ ..\ \max \operatorname{dom} \mathcal{P}]}$; one is returned as the minimum hitting set.

The algorithm tries, as a first attempt, the full integer interval from 0 to $\max \operatorname{dom} \mathcal{P}$ which obviously hits the pattern domain's circular repetition. Then, at any stage, it tests for hitting the next set with lesser cardinality than the current best, according to the aforementioned order.

Reduction to set cover and approximated solution The minimum hitting set problem is equivalent to the *set cover* problem: given a collection $\mathcal{S} \subseteq 2^{[1\ ..\ n]}$, find a subset of $\mathcal{S}$ of minimal cardinality whose union is $[1\ ..\ n]$.

Given a collection $\mathcal{R} \triangleq r_1 \ldots r_n$ of sets, whose union is $[1\ ..\ k]$, the minimum hitting set problem for $\mathcal{R}$ can be formulated as a set cover problem as follows. For each $e \in [1\ ..\ k]$, let $s_e \triangleq \{i \mid e \in r_i\}$, and let $\mathcal{S}$ be the collection of all such sets. The union of each subset $\mathcal{T}$ of $\mathcal{S}$ is the set of indices of the sets in $\mathcal{R}$ which contain one of the indices of the sets in $\mathcal{T}$. Thus, the indices of a solution $\mathcal{T}$ of a set cover problem for $\mathcal{S}$ form a minimum hitting set for $\mathcal{R}$, because $\mathcal{T}$'s union is $[1\ ..\ n]$, so all elements of $\mathcal{R}$ have non-empty intersection with the set of $\mathcal{T}$'s indices, and the union of no proper subset of $\mathcal{T}$ is $[1\ ..\ n]$, or $\mathcal{T}$ would not be minimal.

The set cover problems has been widely studied in the literature; in particular, approximation algorithms have been proposed for it. A very simple algorithm is the so-called *greedy* algorithm: at each step, it chooses the set with higher cardinality, and it deletes such set's elements from the other sets, until the universe is covered.

Its performance (expressed as the ratio between the cardinality of the output and of the minimal covering collection) is $\ln n - \ln \ln n + \Theta(1)$ [17]: in particular, it will not be worse than $\ln n - \ln \ln n + 0.78$, and it cannot be guaranteed to be better than $\ln n - \ln \ln n - 0.31$.

Several inapproximability results for set cover have been shown: to the best of our knowledge, the best lower bound for the performance ratio has been proved by Alon et al. [3], who proved that set cover cannot be approximated efficiently to less than $c \ln n$, for a constant c, unless $P = NP$. Therefore, the greedy algorithm performance is close to the best we can hope for, among polynomial algorithms.

In our case, for the aforementioned reduction of minimum hitting set to set cover, given a natural pattern $\mathcal{P}$, n is max dom $\mathcal{P} + 1$.

$\lvert\text{dom } \mathcal{P}\rvert$	Ratio (average)	Ratio (max)	Time (complete)	Time (greedy)
2	1.0	1.0	24777.18	2.491
4	1.033	1.333	2023.496	1.915
6	1.02	1.2	296.656	1.83
8	1.0	1.0	48.383	1.73
10	1.033	1.333	21.473	1.787
12	1.0	1.0	7.937	1.716
14	1.0	1.0	6.318	1.675
16	1.0	1.0	1.556	1.614
18	1.0	1.0	1.363	1.843
20	1.0	1.0	1.522	1.668

Table 3: Comparison of the complete and greedy algorithms (times are in milliseconds).

Experimental comparison However, in our case, the greedy algorithm performed better experimentally than the theoretical lower bound.

In order to assess performance for our application, we experimented with the two aforementioned solution methods: the complete algorithm for minimum hitting set, and the translation to set cover and solution with the greedy algorithm.

We fixed max dom $\mathcal{P} = 20$; $\lvert\text{dom } \mathcal{P}\rvert$ varies. For each parameter value, we generated a random pattern and computed a covering shape with the two methods; we repeated such each computation 50 times. Results are shown in Table 3. In each line, we show the average and maximum performance ratio, as well as the computation times (on a 2.0GHz dual-core CPU).

As expected, computation time for the complete algorithm grows exponentially with max dom $\mathcal{P} - \lvert\text{dom } \mathcal{P}\rvert$, which makes it inapplicable for bigger problems. The greedy algorithm is obviously faster; and its performance ratio appears to be better than its lower bound which, for $M = 20$, would be $\ln 21 - \ln\ln 21 - 0.31 = 1.62$.

5 Runtime verification of Linear Temporal Logic formulæ

Intuitively, runtime verification is the problem of checking if an execution trace of some system enjoys a property [12]. A common choice for a formalism to express properties is temporal logics and, in particular, Linear Temporal Logics (LTL) [16]. In the following, we discuss runtime verification of (some) LTL formulæ by observing partial execution traces.

5.1 Background

In this section, we briefly recall standard definitions for LTL, and some additional definitions that we use later in the paper.

Definition 13. *Let $\mathcal{A}$ be a countable set of atomic formulæ, $\top$ represent truth and $\bot$ represent falsity. Then a* ***formula*** *is defined recursively as follows:*

- *$\top$ and $\bot$ are formulæ; if $\varphi \in \mathcal{A}$, φ is a formula;*
- *if φ is a formula, $\neg\varphi$, $\mathbf{X}\varphi$, $\Box\varphi$, and $\Diamond\varphi$ are formulæ;*
- *if φ and ψ are formulæ, then $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi\mathbf{U}\psi$ are formulæ.*

A formula's truth value is defined on a *path* in a *Kripke structure*.

Definition 14. *A* ***Kripke structure*** *is a tuple $\langle S, S_0, R, \mathcal{A}, V\rangle$, where S is a finite* ***set of states****, $S_0 \subseteq S$ is a* ***set of initial states****, $R \subseteq S\times S$ is a* ***transition relation****, $\mathcal{A}$ is a countable* ***set of atomic propositions****, and $V : S \to 2^{\mathcal{A}}$ is a* ***labeling function****.*

Definition 15. *Given a Kripke structure $\mathcal{K} \triangleq \langle S, S_0, R, \mathcal{A}, V\rangle$, a* ***path*** *on $\mathcal{K}$ is a sequence of states $\pi \triangleq s_0 s_1 \ldots s_n \ldots$ such that $\forall i \in \mathbb{N} \mid (s_{i+1} \in \pi \Rightarrow \langle s_i, s_{i+1}\rangle \in R)$.*

Definition 16. *Given a path $\pi \triangleq s_0 s_1 \ldots s_n, \ldots$ on a Kripke structure $\mathcal{K}$, the k-th* ***postfix*** *of π is the sequence π^k of states $s_k s_{k+1} \ldots s_n, \ldots$.*

Definition 17. *For LTL formulæ φ and ψ, a Kripke structure $\mathcal{K} \triangleq \langle S, S_0, R, \mathcal{A}, V\rangle$ and a path $\pi \triangleq s_0 s_1 \ldots s_n$ in $\mathcal{K}$,*

- *if $\varphi \in \mathcal{A} \cup \{\top, \bot\}$, $\pi \models \varphi$* iff *$\varphi \in V(s_0)$ or $\varphi = \top$;*
- *$\pi \models \varphi \vee \psi$* iff *$\pi \models \varphi$ or $\pi \models \psi$; $\pi \models \varphi \wedge \psi$* iff *$\pi \models \varphi$ and $\pi \models \psi$; $\pi \models \neg\varphi$* iff *$\pi \not\models \varphi$*

- $\pi \models \mathbf{X}\varphi$ iff $\pi^1 \models \varphi$
- $\pi \models \varphi\mathbf{U}\psi$ iff $\pi \models \psi$, *or* $\exists k > 0 \mid (\pi^k \models \psi \wedge \forall i \mid 0 \leq i < k \mid \pi^i \models \varphi)$
- $\pi \models \Diamond\varphi$ iff $\exists k \geq 0 \mid \pi^k \models \varphi$
- $\pi \models \Box\varphi$ iff $\forall k \geq 0 \mid \pi^k \models \varphi$

Definition 18. *Given a state s in a Kripke structure and a formula φ, $s \models \varphi$ if and only if there exists a path in $\mathcal{K}$ whose first state is s and $\pi \models \varphi$.*

5.2 Partial paths

In this section, we define partial paths, which formalize the idea of incomplete execution traces.

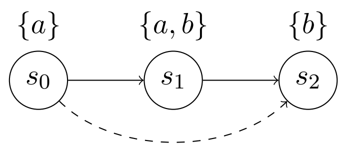
Definition 19. *Given a Kripke structure $\mathcal{K}$, a* ***partial path*** *on $\mathcal{K}$ is a sub-sequence η of some path π on $\mathcal{K}$; η is a* ***sub-path*** *of π ($\eta \sqsubset \pi$).*

Definition 20. *Given a Kripke structure $\mathcal{K}$, a path $\pi \triangleq s_0 s_1 \ldots s_n \ldots$ on $\mathcal{K}$, and a finite set $\mathcal{I}$ of natural numbers whose maximum is M, π's* ***sampling*** *of shape $\mathcal{I}$, $\pi_{\mathcal{I}}$, is the sequence of states s_i such that s_i is a state in π and $i = q + k(M+1)$, for some $k \in \mathbb{N}$ and $q \in \mathcal{I}$.*

Example 7. *The sampling of a path $\pi \triangleq s_0 s_1 \ldots s_{16}$ of shape $\{0, 1, 3, 6\}$ is $s_0 s_1 s_3 s_6 s_7 s_8 s_{10} s_{13} s_{14} s_{15}$.*

Graphical representation of paths A path is represented graphically as follows: a state s is represented as a node with label s; the set of atomic propositions satisfied by each state is written above the corresponding node; solid edges represent the sequence in the full path, and dashed edges represent the sequence in the partial path.

For example, in this representation the complete path is composed of the states s_0, s_1 and s_2, while the partial path is composed of the states s_0 and s_2. State s_0 satisfies the atomic proposition a, s_1 satisfies a and b, and s_2 satisfies b.



5.3 Semantics in partial paths

In general, a partial path is not a path in the original Kripke structure (because two consecutive states are not in the transition relation). Therefore, in order to give semantics to formulæ on the partial path, an associated Kripke structure can be defined as follows.

Definition 21. *Let $\mathcal{K} \triangleq \langle S, S_0, R, \mathcal{A}, V\rangle$ be a Kripke structure and π a path in $\mathcal{K}$. Then, for each partial path $\eta \triangleq s_0 s_1 \ldots s_n$, with $\eta \sqsubset \pi$, the* ***associated Kripke structure*** *$\mathcal{K}_\eta \triangleq \langle S_\eta, S_{0\eta}, R_\eta, \mathcal{A}_\eta, V_\eta\rangle$ is defined, where S_η is the set of states in η, $S_{0\eta} \triangleq \{s_0\}$, $R_\eta \triangleq \{\langle s_i, s_{i+1}\rangle \mid 0 \leq i < n\}$, $\mathcal{A}_\eta \triangleq \mathcal{A}$, and V_η is the restriction of V to S_η.*

The semantics of formulæ on a partial path can then be defined as in Defs. 17 and 18, where the relevant Kripke structure is the one associated with the partial path.

5.4 Formula entailment in partial and full paths

In this section, we discuss logical relations between entailment of a LTL formula in full and partial paths.

Box operator ($\Box$)

Theorem 3. *If $\eta \sqsubset \pi$, then $\pi \models \Box\varphi \Rightarrow \eta \models \Box\varphi$.*

Proof. Let s be a state in η; then, by $\eta \sqsubset \pi$, $s \in \pi$. By hypothesis, $s \models \varphi$. The same holds for any state in η. $\Box$

The opposite does not hold:

$$\begin{tikzcd}
\{a\} & \{\} & \{a\} \\
s_0 \arrow[r] \arrow[rr, dashed, bend right] & s_1 \arrow[r] & s_2
\end{tikzcd}$$

$\eta \models \Box a$, but $\pi \not\models \Box a$.

Diamond operator ($\Diamond$)

Theorem 4. *If $\eta \sqsubset \pi$, then $\pi \models \Diamond\varphi \Leftarrow \eta \models \Diamond\varphi$.*

Proof. Let s be the state in η such that $s \models \varphi$(such a state exists by hypothesis). Since $\eta \sqsubset \pi$, $s \in \pi$; i.e., $\exists s \in \pi \mid s \models \varphi$. $\Box$

The opposite does not hold:

$$\begin{tikzcd}
\{a\} & \{a,b\} & \{a\} \\
s_0 \arrow[r] \arrow[rr, dashed, bend right] & s_1 \arrow[r] & s_2
\end{tikzcd}$$

$\pi \models \Diamond b$, but $\eta \not\models \Diamond b$.

Next operator (X) No general result can be proved about **X**.

$$\begin{tikzcd}
\{\} & \{a\} & \{b\} \\
s_0 \arrow[r] \arrow[rr, dashed, bend right] & s_1 \arrow[r] & s_2
\end{tikzcd}$$

- $\pi \models \mathbf{X}a$, but $\eta \not\models \mathbf{X}a$
- $\eta \models \mathbf{X}b$, but $\pi \not\models \mathbf{X}b$.

Until operator (U) In general, no implication can be proved about **U**.

$$\begin{tikzcd}
\{a,b\} & \{b\} & \{b,c\} & \{d\} \\
s_0 \arrow[r] \arrow[rr, dashed, bend right] & s_1 \arrow[r] & s_2 \arrow[r] & s_3
\end{tikzcd}$$

- $\pi \models b\mathbf{U}d$, but $\eta \not\models b\mathbf{U}d$
- $\eta \models a\mathbf{U}c$, but $\pi \not\models a\mathbf{U}c$

5.5 Event detection in LTL

The results in Section 5.4 are too weak to be of practical use. However, in an application where (i) it is of interest to detect if a formula φ is entailed by at least one state in a path (that is, to verify $\Diamond\varphi^1$), and (ii) it is known that φ's truth follows a given pattern (e.g., φ is entailed by a number of consecutive states, or it is entailed by a state, then by the following state, then by the state that comes after four more states), the result of Theorem 1 can be applied.

First, we define formally what it means for a LTL formula's truth to follow a given (natural) pattern.

Definition 22. *Given a natural pattern $\mathcal{P}$, a formula φ is an **LTL-event** of pattern $\mathcal{P}$ in a Kripke structure $\mathcal{K}$ if and only if for any path $\pi \triangleq s_0 s_1 \dots s_n$ in $\mathcal{K}$, $(\exists k \mid s_k \models \varphi) \Rightarrow \exists i \mid (\forall j \in \operatorname{dom}\mathcal{P} \mid s_{i+j} \models \varphi)$.*

The following results allows to verify the occurrence of a LTL formula by observing a sampling of a path, rather than the full path.

Theorem 5. *Let φ be a LTL-event of pattern $\mathcal{P}$ on a Kripke structure $\mathcal{K}$. Let $\mathcal{S}_C$ be a covering shape of $\mathcal{P}$'s. Then, for each path $\pi \triangleq s_0 s_1 \dots s_n \dots$ on $\mathcal{K}$, $\pi \models \Diamond\varphi$ if and only if $\pi_{\mathcal{S}_C} \models \Diamond\varphi$.*

Proof. If: by Theorem 4, because $\pi_{\mathcal{S}_C}$ is a subpath of π.

Only if: $\pi \models \Diamond\varphi$, so $\exists k \mid s_k \models \varphi$. By Definition 22, $\exists i \mid (\forall j \in \operatorname{dom}\mathcal{P} \mid s_l \models \varphi, l = i + j)$. Such ls are the observability set of an event of pattern $\mathcal{P}$ and offset i (see Definition 5). Since $\mathcal{S}_C$ is $\mathcal{P}$'s covering shape, by Theorem 1 the set $\mathcal{C} \triangleq \{q + k(M+1) \mid q \in \mathcal{S}_C \wedge k \in \mathbb{N}\}$, where $M \triangleq \max\mathcal{S}_C$, intersects the observability sets of all the events of shape $\mathcal{P}$, that is, it contains at least one of the aforementioned ls.

Summarizing, there exists some l with the following properties: (i) $s_l \in \pi$, (ii) $l = q + k(M+1)$, for some $q \in \mathcal{S}_C$ and $k \in \mathbb{N}$, and (iii) $s_l \models \varphi$. But properties (i) and (ii) define indices of states that belong to $\pi_{\mathcal{S}_C}$, so $s_l \in \pi_{\mathcal{S}_C}$; and because of property (iii), $\pi_{\mathcal{S}_C} \models \Diamond\varphi$.

□

[1] If the formula is understood as a logical represenetation of an event, this amounts to verifying the event's occurrence.

6 Discussion

In this paper, we focussed on one-dimensional natural patterns. A first generalization that comes to mind is to consider multi-dimensional natural coordinate systems.

Another generalization is when events with different patterns can occur. In this case, we can distinguish two cases: one where the value sets of the patterns are disjoint (so an event can be immediately recognized), and the other where, once an event is detected, more observations are required in order to determine its pattern.

If more than one event is considered, complex events can be composed from simple ones by conjunction, disjunction, sequence or negation (as in event algebras [4]).

This brings out the opportunity to employ model-based diagnosis techniques in general, namely efficient kernel diagnosis algorithms that detect maximal intersections of hitting sets [6], and allow for detection of multiple persistent and intermittent diagnosis[5].

In case detection of all events is not critical, one may consider further reducing the number of observations, and estimate the probability of missing an event; decision theory can be applied to find a trade-off between the costs of observation and of failed detection.

Tools and techniques developed in the field of logic programming can be used to go beyond event detection, in case of partial execution traces. In particular, approaches based on abductive logic programming [10] and its efficient implementation [2] appear appropriate, as abduction can be used to formulate hypotheses on the state of the system in the time points not subject to observation.

A first extension that comes to mind is to consider more than one type of event. The occurrence of an event would be represented by an abducible predicate. For instance, it can be of interest to express permissible combinations of events, which may be expressed by means of integrity constraints; or to express relations of expectancy of an event depending on the detection of another.

For instance, the $\mathcal{S}$CIFF abductive logic language [1] supports special predicate $\mathbf{H}(a)$, meaning that event a happened, and abducible predicates $\mathbf{E}(a)$ and $\mathbf{EN}(a)$, meaning, respectively, that event a is expected to happen, or expected not to happen. Integrity constraints as $\mathbf{H}(a) \Rightarrow \mathbf{E}(b) \vee \mathbf{E}(c)$ can be used as a guidance mechanism to perform detection: a's occurrence would trigger an attempt to detect the occurrence of b or c, in order to satisfy the integrity constraint.

More generally, events sometimes have tell-tale side-effects, but which are not guaranteed to happen – i.e., the side-effects either don't always follow if the event happens, or they may be side-effects of only a subset of the whole event, so that while the side-effects may actually happen (and even then with a probability) the complete event might not be there.

Also, if detected they may assure us the event occurred (though not being formally part of the event) or simply allow to hypothesize that event or some other event. The classical example in this case, framed using abduction, is "the grass is wet": did it rain during the night or was the sprinkler left on or both? On the other hand, it may have rained but the grass no longer being wet.

These tell-tale side-effects can be used heuristically as triggers to direct attention and speed up event detection by going for the more likely events first, given the side-effects. A Bayes net could be employed to code such heuristic and probabilistic information. In particular, side-effects can be tell-tale signs of intentions, which in turn lead to actions and events. This connects the event detection issues to intention recognition, combining Bayes nets with plan recognition, and combining uncertainty KR and rule KR in general [14, 13]. [11] reports on a logic programming based system which infers team of agents' intentions from traces of agents' action events and observations.

Similar to side-effects, in a mirror-like way, one may have premonitions, that is tell-tale prior-effects that may be omens of (sub-)events to follow, for which all of the above can likewise be re-stated, with consequences for heuristics leading to faster or priority event recognition.

The importance of the tell-tale signs, prior or posterior, can be evaluated for significance and preferential compatibility. A case in point might be the tell-tale signs concerning a possible natural disaster or deliberate attack, with consequences for preventive or palliative measures. This raises the issue of how can events be countered or forestalled. On the other hand, positive events might be leveraged for greater benefit.

In case execution traces are being observed, but there is uncertainty about observations, predictive lookahead can be used to remove uncertainty. Same goes, regarding uncertainty removal, for retrospective hindsight (by making observations about the past, cf. astronomy) in order to confirm or disconfirm hypotheses about traces.

If the system being discussed can be controlled (such as a logic program that can be evolved [7]), the problem can be viewed in the other direction, i.e., how to update the system in order to have it generate the desired traces.

Such problems can have more than one solution. Preferences (now well understood in the context of logic programming [15]) can be used to rank among solutions, or as a heuristics along with abduction or evolution [8].

Thanks to the recent advances in logic programming technology, one may also think about trying to confirm and disconfirm an event at the same time (using multi-threaded computations) and take decisions depending on the first goal that succeeds.

In summary, the topic of runtime verification and event detection can benefit from declarative logic programming because of the natural use of logic programming, abduction, and event calculus for representing observations and actions, as well as and its implementation tools in general, in order to support the topic's ramifications and applications. The latter two concern not just opening declarative programming to the new areas of monitoring and control, but also the very use of runtime verification in program correctness and runtime safety.

7 Conclusions

In this paper, we considered the problem of reducing the number of observations for reliable event detection. We showed that, when the observation space can be modeled as the set of natural numbers, the number of observations can be significantly reduced. In this case, the problem can be reduced to a NP-hard problem, but an approximation algorithm with good (theoretical and experimental) performance can be applied. We showed how the result can be applied to runtime verification of temporal logic formulæ. We also discussed possible extensions and generalizations of the problem, and argued for the use of logic programming techniques and systems to handle increased sophistication.

References

[1] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable agent interaction in abductive logic programming: the $\mathcal{S}$CIFF framework. *ACM Transactions on Computational Logic (TOCL)*, 9(4), 2008.

[2] J. J. Alferes, L. M. Pereira, and T. Swift. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *Theory and Practice of Logic Programming*, 4(4):383–428, 2004.

[3] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.

[4] J. Carlson and B. Lisper. An event detection algebra for reactive systems. In *Proceedings of the 4th ACM international conference on Embedded software*, pages 147–154, New York, NY, USA, 2004. ACM.

[5] J. de Kleer. Diagnosing multiple persistent and intermittent faults. In C. Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 733–738, 2009.

[6] J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artif. Intell.*, 56(2-3):197–222, 1992.

[7] P. Dell'Acqua, A. Lombardi, and L. M. Pereira. Modelling adaptive controllers with evolving logic programs. In J. Andrade-Cetto, J.-L. Ferrier, J. D. Pereira, and J. Filipe, editors, *ICINCO 2006, Setúbal, Portugal*. INSTICC Press, 2006.

[8] P. Dell'Acqua and L. M. Pereira. Preferential theory revision. *J. Applied Logic*, 5(4):586–601, 2007.

[9] A. Hinze, K. Sachs, and A. Buchmann. Event-based applications and enabling technologies. In *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1–15, New York, NY, USA, 2009. ACM.

[10] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[11] T. Kanno, K. Nakata, and K. Furuta. A method for team intention inference. *Int. J. Hum.-Comput. Stud.*, 58(4):393–413, 2003.

[12] M. Leucker and C. Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293 – 303, 2009.

[13] L. M. Pereira and H. T. Anh. Intention recognition via causal bayes networks plus plan generation. In L. S. Lopes, N. Lau, P. Mariano, and L. Rocha, editors, *14th Portuguese Intl.Conf. on Artificial Intelligence*, volume 5816 of *LNAI*, pages 138–149. Springer, October 2009.

[14] L. M. Pereira and H. T. Anh. Intention recognition with evolution prospection and causal bayesian networks. In A. Madureira, J. Ferreira, and Z. Vale, editors, *Computational Intelligence for Engineering Systems 3: Emergent Applications*, Intelligent Systems, Control and Automation: Science and Engineering Book Series. Springer, 2010. To appear.

[15] L. M. Pereira, P. Dell'Acqua, and G. Lopes. On preferring and inspecting abductive models. In A. Gill and T. Swift, editors, *Practical Aspects of Declarative Languages, 11th International Symposium, PADL 2009, Savannah, GA, USA, January 19-20, 2009. Proceedings*, volume 5418 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2009.

[16] A. Pnueli. The temporal logics of programs. In *Proceedings of the 18th Symposium on the Foundations of Computer Science*, pages 46–57, 1977.

[17] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 435–441, New York, NY, USA, 1996. ACM.