# A graph-theoretic approach to efficiently reason about partially ordered events in the Event Calculus

**Massimo Franceschet** **Angelo Montanari**
Dipartimento di Matematica e Informatica, Università di Udine
Via delle Scienze, 206 – 33100 Udine, Italy
$\{$*francesc*$|$*montana*$\}$*@dimi.uniud.it*

## Abstract

*In this paper, we exploit graph-theoretic techniques to efficiently reason about partially ordered events in the Event Calculus. We replace the traditional generate-and-test reasoning strategy by a more efficient generate-only one that operates on the underlying directed acyclic graph of events representing ordering information by pairing breadth-first and depth-first visits in a suitable way. We prove the soundness and completeness of the proposed strategy, and thoroughly analyze its computational complexity. Furthermore, we show how it can be generalized to deal with the Modal Event Calculus, that provides a uniform modal framework for the basic Event Calculus and its skeptical and credulous variants.*

## 1. Introduction

In this paper, we propose a graph-theoretic approach to the problem of efficiently reasoning about partially ordered events in Kowalski and Sergot's Event Calculus (*EC* for short) [5, 11] and in its skeptical and credulous modal variants [4]. Reasoning about the evolution of the world as the result of the occurrence of a set of events is crucial in a variety of applications, including diagnosis, robotics, agent modelling, qualitative physics, monitoring, planning and plan validation, and natural language understanding. In many of these applications, a reasoner is forced to deal with incomplete knowledge about the events it is concerned with and/or their temporal order [9]. We consider the problem of efficiently inferring what is true over certain event-bounded time intervals when only incomplete knowledge is available. Even though we develop our solution in the (Modal) Event Calculus framework, we expect it to be applicable to any formalism for reasoning about partially ordered events.

Partial ordering information about event occurrences can be naturally represented by means of a directed acyclic graph $G = \langle E, o\rangle$, where the set of nodes $E$ is the set of events and, for every $e_i, e_j \in E$, there exists $(e_i, e_j) \in o$ if and only if it is known that $e_i$ occurs before $e_j$. *EC* updates are of additive nature only and they just consist in the addition of new events ($G$ nodes) and/or of further (consistent) ordering information about the given events ($G$ edges). Given a directed acyclic graph $G = \langle E, o\rangle$, representing a set of partially ordered events, *EC* allows one to compute the set of event-bounded maximal validity intervals (*MVI*s for short) over which the properties initiated and terminated by such events hold uninterruptedly. To compute the set of *MVI*s for any given property $p$, it exploits a simple *generate-and-test* strategy [5]: first, it blindly picks up all pairs $(e_i, e_t)$ of initiating and terminating events for $p$; then, it checks whether or not they occur in the proper ordering, that is, if the initiating event $e_i$ precedes the terminating event $e_t$; finally, it looks for possible interrupting events $e$ occurring in between. Checking whether $e_i$ precedes $e_t$ or not reduces to establish if the edge $(e_i, e_t)$ belong to the transitive closure $o^+$ of $o$ as well as checking if there exists an interrupting event $e$ for $p$ in $(e_i, e_t)$ requires to verify if both $(e_i, e)$ and $(e, e_t)$ belong to $o^+$. In [6], Chittaro et alii have shown that the complexity of query processing based on this simple generate-and-test strategy is $\mathcal{O}(n^5)$, provided that suitable graph marking techniques are used.

In this paper, we propose a more efficient *generate-only* strategy which reduces the computation of the set of *MVI*s for any given property $p$ to a non-standard visit of the graph $G$. The idea of exploiting graph-theoretic techniques to speed up temporal reasoning about partially ordered events in *EC* was originally proposed by Chittaro et alii in [6]. They provide a precise characterization of what *EC* actually does to compute *MVI*s and show that, whenever all recorded events are concerned with the same unique property $p$, shifting the perspective from the transitive closure $o^+$ of the given partial order $o$ to its transitive reduction $o^-$ allows one to do it more efficiently using a generate-only strategy. Their solution can be easily generalized to the case of multiple incompatible properties, that is, properties whose validity intervals cannot overlap. In this paper, we will first show that such a solution cannot be further extended to deal with the general multiple-property case, because it does not properly work whenever there exist two or more non-transitive paths of different length between an ordered pair of events that respectively initiate and terminate a given property. Then, we will describe an alternative *generate-only* strategy for *MVI*s computation in the general multiple-property case, which pairs breadth-first and depth-first visits of $o$ in a suitable way, and thoroughly analyze its complexity.

As pointed out in [5], however, when only partial information about the occurred events and their exact order is available, the sets of *MVI*s derived by *EC* bear little relevance, since the acquisition of additional knowledge about the set of events and/or their occurrence times might both dismiss current *MVI*s and validate new *MVI*s. To overcome these limitations, two variants of the basic *EC*, respectively called the *Skeptical EC* (*SKEC*) and the *Credulous EC* (*CREC*), have been proposed in [7]. For any given property $p$, *SKEC* computes the set of necessarily true *MVI*s, that is, the set of *MVI*s which are derivable in all refinements of the given partial order, while *CREC* computes the set of possibly true *MVI*s, that is, the set of *MVI*s which are derivable in at least one refinement of the given partial order. In [2], Cervesato et alii defined a uniform modal interpretation for *EC*, *SKEC*, and *CREC*, called the *Modal Event Calculus* (*MEC*), and extended the generate-and-test strategy for *MVI*s computation in *EC* to *MEC*, without any rise in computational complexity [3]. In the last part of the paper, we will show that the proposed generate-only strategy for *MVI*s computation in *EC* can be easily tailored to *MEC*.

The paper is organized as follows. In Section 2, we recall some background knowledge on ordering relations, transitive reduction, and transitive closure. In Section 3, we present the basic features and properties of *EC* and *MEC*, and point out the limitations of the existing algorithms for *MVI*s computation when only partial ordering information is available. In Section 4, we describe a new generate-only algorithm for *MVI*s computation and prove its soundness and completeness. The increase in efficiency of the proposed solution is demonstrated by the complexity analysis reported in Section 5. In Section 6, we show how to adapt the proposed algorithm to *MEC*. In the conclusions, we briefly discuss the achieved results and outline possible directions for future research.

## 2. On ordering relations, transitive reduction, and transitive closure

Let us first remind some basic notions about ordering relations and ordered sets, transitive closure, and transitive reduction upon which we will rely in the following [12].

*EC* usually *represents* ordering information as a binary acyclic relation on the set of events, that is, as an ordering relation possibly missing some transitive links, but it *uses* ordering information as a (strict) partial order that can be recovered as the transitive closure of the given binary acyclic relation.

**Definition 2.1** (*DAGs, strictly ordered sets, non-strictly ordered sets, generated DAGs, induced DAGs*)

*Let $E$ be a set and $o$ a binary relation on $E$. $o$ is called a (strict)* partial order *if it is irreflexive and transitive (and, thus, asymmetric), while it is called a* reflexive partial order *if it is reflexive, antisymmetric, and transitive. The pair $(E, o)$ is called a* directed acyclic graph *(*DAG*) if $o$ is a binary acyclic relation; a* strictly ordered set *if $o$ is a partial order; a* non-strictly ordered set *if $o$ is a reflexive partial order. Moreover, given a DAG $G = \langle E, o\rangle$ and a node $e \in E$, the subgraph $G(e)$ of $G$ consisting of all and only the nodes which are accessible from $e$ and of the edges that connect them is called the graph* generated by $e$. *Finally, given a DAG $G = \langle E, o\rangle$ and a set $T \subseteq E$, the subgraph of $G$* induced by $T$ *consists of the nodes in $T$ and the subset of edges in $o$ that connect them.* □

We will denote the sets of all binary acyclic relations and of all partial orders on $E$ as $O_E$ and $W_E$, respectively. It is easy to show that, for any set $E$, $W_E \subseteq O_E$. Moreover, we will use the letters $o$ and $w$, possibly subscripted, to denote binary acyclic relations and partial orders, respectively.

When one is mainly interested in representing the path information of a *DAG*, two extreme approaches can be followed: (i) *transitive reduction* (or minimum storage representation), and (ii) *transitive closure* (or minimum query time representation). Transitive reduction and closure of a *DAG* can be formally defined as follows.

**Definition 2.2** (*Transitive reduction and closure of DAGs*)

*Let $G = \langle E, o\rangle$ be a directed acyclic graph. The* transitive reduction *of $G$ is the (unique) graph $G^- = \langle E, o^-\rangle$ with the smallest number of edges, with the property that, for any pair of nodes $i, j \in E$, there is a directed path from $i$ to $j$ in $G$ if and only if there is a directed path from $i$ to $j$ in $G^-$. The* transitive closure *of $G$ is the (unique) graph $G^+ = \langle E, o^+\rangle$ with the property that, for any pair of nodes $i, j \in E$ there is a directed path $i$ to $j$ in $G$ if and only if there is an edge $(i, j) \in o^+$ in $G^+$.* □

In [1], Aho et alii show that every (directed) graph has a transitive reduction, which can be computed in polynomial time. They also show that such a reduction is unique in the case of directed acyclic graphs. Furthermore, they prove that the time needed to compute the transitive reduction of a graph differs from the time needed to compute its transitive closure by at most a constant factor.

*MVI*s computation requires the derivation of the transitive closure of the given partial order. Clearly, if $(E, o)$ is a *DAG*, then $(E, o^+)$ is a strictly ordered set. We say that two binary acyclic relations $o_1, o_2 \in O_E$ are *equally informative* if $o_1^+ = o_2^+$. This induces an equivalence relation $\sim$ on $O_E$. It is easy to prove that, for any set $E$, the quotient set $O_E/\sim$ and $W_E$ are isomorphic. In the following, we will often identify a binary acyclic relation $o$ with the corresponding element $o^+$ of $W_E$.

## 3. Basic and Modal Event Calculi

In this section, we first recall the syntax and semantics of *EC* and *MEC*; then, we discuss the effects of the addition of new events and/or pieces of ordering information on the sets of *MVI*s computed by *EC* and *MEC*; finally, we briefly review the existing algorithms for *MVI*s computation.

### 3.1. Syntax and semantics of EC and MEC

Kowalski and Sergot's *Event Calculus* (*EC*) [11] aims at modeling situations that consist of a set of events, whose occurrences over time have the effect of initiating or terminating the validity of properties, some of which may be mutually exclusive. We formalize the time-independent aspects of a situation by means of an *EC-structure*, which is defined as follows [4].

**Definition 3.1** (*EC-structure*)

*A* structure *for the* Event Calculus *(abbreviated* EC-structure*) is a quintuple* $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ *such that:*

- $E = \{e_1, \ldots, e_n\}$ *and* $P = \{p_1, \ldots, p_m\}$ *are finite sets of* events *and* properties*, respectively;*
- $[\cdot\rangle : P \to \mathbf{2}^E$ *and* $\langle\cdot] : P \to \mathbf{2}^E$ *are respectively the* initiating *and* terminating map *of* $\mathcal{H}$*. For every property* $p \in P$*,* $[p\rangle$ *and* $\langle p]$ *represent the set of events that initiate and terminate* $p$*, respectively;*
- $]\cdot,\cdot[ \subseteq P \times P$ *is an irreflexive and symmetric relation, called the* exclusivity relation*, that models exclusivity among properties.* □

Unlike the original presentation of *EC* [11], we focus our attention on situations where the occurrence time of events is unknown. Indeed, we assume that incomplete information about the relative order in which events occur is available. We however require temporal data to be consistent so that an event cannot both precede and follow any other event. We formalize the time-dependent aspects of an *EC* problem by specifying a partial order, called *knowledge state*, on the set of events $E$ [4].

Given a structure $\mathcal{H}$, we adopt as the *query language* of *EC* the set:

$$\mathcal{L}(\mathrm{EC}) = \{p(e_1, e_2) : p \in P \text{ and } e_1, e_2 \in E\}$$

of all property-labeled intervals over $\mathcal{H}$.

Given a knowledge state $w$, a *maximal validity interval* (*MVI*) for a property $p$ with respect to $w$ is an interval of $w$ over which the property $p$ holds uninterruptedly. We represent an *MVI* for $p$ as $p(e_i, e_t)$, where $e_i$ and $e_t$ are the events that initiate and terminate the interval over which $p$ maximally holds, respectively. The task performed by *EC* reduces to deciding which of the elements of $\mathcal{L}(\mathrm{EC})$ are *MVI*s and which are not, with respect to the current partial order of events.

We interpret the elements of $\mathcal{L}(\mathrm{EC})$ relative to the set $W_E$ (denoted $W_{\mathcal{H}}$ in this context) of partial orders among events in $E$. In order for $p(e_1, e_2)$ to be an *MVI* relative to the knowledge state $w$, $(e_1, e_2)$ must be an interval in $w$, i.e. $e_1 <_w e_2$. Moreover, $e_1$ and $e_2$ must witness the validity of the property $p$ at the ends of this interval by initiating and terminating $p$, respectively. These requirements are enforced by conditions *i.*, *ii.* and *iii.*, respectively, in the definition of valuation given below. The maximality requirement is caught by the negation of the meta-predicate $br(p, e_1, e_2, w)$ in condition *iv.*, which expresses the fact that the truth of an *MVI* must not be *broken* by any interrupting event. Any event $e$ which is known to have happened between $e_1$ and $e_2$ in $w$ and that initiates or terminates a property that is either $p$ itself or a property exclusive with $p$ interrupts the truth of $p(e_1, e_2)$. These observations are formalized as follows [4].

**Definition 3.2** (*Intended model of EC*)

*Let* $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ *be a EC-structure. The* intended EC-model *of* $\mathcal{H}$ *is the propositional valuation* $\upsilon_{\mathcal{H}} : W_{\mathcal{H}} \to 2^{\mathcal{L}(\mathrm{EC})}$*, where* $\upsilon_{\mathcal{H}}$ *is defined in such a way that* $p(e_1, e_2) \in \upsilon_{\mathcal{H}}(w)$ *if and only if*

***i.*** $e_1 <_w e_2$*;*

***ii.*** $e_1 \in [p\rangle$*;*

***iii.*** $e_2 \in \langle p]$*;*

***iv.*** $br(p, e_1, e_2, w)$ *does not hold, where* $br(p, e_1, e_2, w)$ *abbreviates*

> there exists an event $e \in E$ such that $e_1 <_w e$, $e <_w e_2$, and there exists a property $q \in P$ such that $e \in [q\rangle$ or $e \in \langle q]$, and either $]p, q[$ or $p = q$.

□

As a general rule, an event $e$ interrupts the validity of a property $p$ if it initiates or terminates $p$ itself or a property $q$ which is incompatible with $p$. This rule adopts the so-called *strong interpretation* of the initiate and terminate relations: given a pair of events $e_i$ and $e_t$, with $e_i$ occurring before $e_t$, that respectively initiate and terminate a property $p$, we conclude that $p$ does not hold over $(e_i, e_t)$ if an event $e$ which initiates or terminates $p$, or a property incompatible with $p$, occurs during this interval, that is, $(e_i, e_t)$ is a candidate *MVI* for $p$, but $e$ forces us to reject it. The strong interpretation is needed when dealing with incomplete sequences of events or incomplete information about their ordering. An alternative interpretation of the initiate and terminate relations, called *weak interpretation*, is also possible. According to such an interpretation, a property $p$ is initiated by an initiating event unless it has been already initiated and not yet terminated (and dually for terminating events). Further details about the strong/weak distinction can be found in [4].

In the case of partially ordered events, the set of *MVI*s derived by *EC* is not stable with respect to the acquisition of new ordering information. Indeed, if we extend the current partial order with new pairs of events, current *MVI*s might become invalid and new *MVI*s can emerge. The *Modal Event Calculus* (*MEC*) allows one to identify the set of MVIs that cannot be invalidated no matter how the ordering information is updated, as far as it remains consistent, and the set of event pairs that will possibly become MVIs depending on which ordering data are acquired. These two sets are called *necessary MVIs* and *possible MVIs*, respectively, using $\Box$-*MVIs* and $\Diamond$-*MVIs* as abbreviations.

The query language $\mathcal{L}(\text{MEC})$ of *MEC* consists of formulas of the form $p(e_1, e_2)$, $\Box p(e_1, e_2)$ and $\Diamond p(e_1, e_2)$, for every property $p$ and events $e_1$ and $e_2$ defined in $\mathcal{H}$.

The intended model of *MEC* is given by shifting the focus from the current knowledge state to all knowledge states that are accessible from it. Since $\subseteq$ is a reflexive partial order, $(W_{\mathcal{H}}, \subseteq)$ can be naturally viewed as a finite, reflexive, transitive and antisymmetric modal frame. This frame, together with the straightforward modal extension of the valuation $\upsilon_{\mathcal{H}}$ to the transitive closure of an arbitrary knowledge state, provides a modal model for *MEC*.

**Definition 3.3** (*Intended model of MEC*)

*Let $\mathcal{H}$, $W_{\mathcal{H}}$, and $\upsilon_{\mathcal{H}}$ be defined as in Definition 3.2. The* MEC-frame *$\mathcal{F}_{\mathcal{H}}$ of $\mathcal{H}$ is the frame $(W_{\mathcal{H}}, \subseteq)$. The* intended MEC-model *of $\mathcal{H}$ is the modal model $\mathcal{I}_{\mathcal{H}} = (W_{\mathcal{H}}, \subseteq, \upsilon_{\mathcal{H}})$. Given $w \in W_{\mathcal{H}}$ and $\varphi \in \mathcal{L}(\text{MEC})$, the truth of $\varphi$ at $w$ with respect to $\mathcal{I}_{\mathcal{H}}$, denoted by $\mathcal{I}_{\mathcal{H}}; w \models \varphi$, is defined as follows:*

$$
\begin{array}{lll}
\mathcal{I}_{\mathcal{H}}; w \models p(e_1, e_2) & \textit{iff} & p(e_1, e_2) \in \upsilon_{\mathcal{H}}(w); \\
\mathcal{I}_{\mathcal{H}}; w \models \Box p(e_1, e_2) & \textit{iff} & \forall w' \,.\, w' \in W_{\mathcal{H}} \wedge w \subseteq w', \\
 & & \Rightarrow \mathcal{I}_{\mathcal{H}}; w' \models p(e_1, e_2); \\
\mathcal{I}_{\mathcal{H}}; w \models \Diamond p(e_1, e_2) & \textit{iff} & \exists w' \,.\, w' \in W_{\mathcal{H}} \wedge w \subseteq w' \\
 & & \wedge\, \mathcal{I}_{\mathcal{H}}; w' \models p(e_1, e_2).
\end{array}
$$

□

Given an EC-structure $\mathcal{H}$ and a partial order $w$, the sets of *MVI*s that are necessarily and possibly true in $w$ correspond respectively to the $\Box$- and $\Diamond$-moded atomic formulas which are valid in $\mathcal{H}$ with respect to $w$. We define the sets $MVI(\mathcal{H}, w)$, $\Box MVI(\mathcal{H}, w)$ and $\Diamond MVI(\mathcal{H}, w)$ of respectively *MVI*s, necessary *MVI*s and possible *MVI*s which a true in $\mathcal{H}$ with respect to $w$ as follows:

$$
\begin{array}{lll}
MVI(\mathcal{H}, w) & = & \{p(e_1, e_2) : I_{\mathcal{H}}; w \models p(e_1, e_2)\}; \\
\Box MVI(\mathcal{H}, w) & = & \{p(e_1, e_2) : I_{\mathcal{H}}; w \models \Box p(e_1, e_2)\}; \\
\Diamond MVI(\mathcal{H}, w) & = & \{p(e_1, e_2) : I_{\mathcal{H}}; w \models \Diamond p(e_1, e_2)\}.
\end{array}
$$

In [2], it has been shown that the sets of $\Box$- and $\Diamond$-MVIs can be computed by exploiting necessary and sufficient *local conditions* over $w$, thus avoiding a complete (and expensive) search of all the consistent refinements of $w$. More precisely, a property $p$ necessarily holds between two events $e_1$ and $e_2$ if and only if the interval $(e_1, e_2)$ belongs to the current order, $e_1$ initiates $p$, $e_2$ terminates $p$, and no event that either initiates or terminates $p$ (or a property incompatible with $p$) will ever be consistently located between $e_1$ and $e_2$. Similarly, a property $p$ may possibly hold between $e_1$ and $e_2$ if and only if the interval $(e_1, e_2)$ is consistent with the current ordering, $e_1$ initiates $p$, $e_2$ terminates $p$, and there are no already known interrupting events between $e_1$ and $e_2$. This is precisely expressed by the following proposition.

**Proposition 3.4** (*Local conditions*)

*Let $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ be a EC-structure. For any atomic formula $p(e_1, e_2)$ on $\mathcal{H}$ and any $w \in W_{\mathcal{H}}$,*

- $\mathcal{I}_{\mathcal{H}}; w \models \Box p(e_1, e_2)$ *if and only if*
  - ***i.*** $(e_1, e_2) \in w$;
  - ***ii.*** $e_1 \in [p\rangle$;
  - ***iii.*** $e_2 \in \langle p]$;
  - ***iv.*** $sbr(p, e_1, e_2, w)$ *does not hold, where* $sbr(p, e_1, e_2, w)$ *abbreviates*

    *there exists an event $e \in E$ such that $(e, e_1) \notin w$, $e \neq e_1$, $(e_2, e) \notin w$, $e \neq e_2$, and there exists a property $q \in P$ such that $e \in [q\rangle$ or $e \in \langle q]$, and either $]p, q[$ or $p = q$.*

- $\mathcal{I}_{\mathcal{H}}; w \models \Diamond p(e_1, e_2)$ *if and only if*
  - ***i.*** $(e_2, e_1) \notin w$;
  - ***ii.*** $e_1 \in [p\rangle$;
  - ***iii.*** $e_2 \in \langle p]$;
  - ***iv.*** $br(p, e_1, e_2, w)$ *does not hold.* ■

Proposition 3.4 allows us to give an alternative definition of the sets $\Box MVI(\mathcal{H}, w)$ and $\Diamond MVI(\mathcal{H}, w)$. Given

$w \in W_{\mathcal{H}}$ and $p \in P$, let $S(\mathcal{H}, w)$ be the set of atomic formulas $p(e_1, e_2)$ such that all other events in $E$ that initiate or terminate $p$, or a property incompatible with $p$, are ordered with respect to $e_1$ and $e_2$ in $w$, and let $C(\mathcal{H}, w)$ be the set of atomic formulas $p(e_1, e_2)$ such that $e_1$ initiates $p$, $e_2$ terminates $p$, and $e_1$ and $e_2$ are unordered in $w$. The set $\Box MVI(\mathcal{H}, w)$ (resp. $\Diamond MVI(\mathcal{H}, w)$) can be alternatively defined as the intersection (resp. union) of the set $MVI(\mathcal{H}, w)$ with $S(\mathcal{H}, w)$ (resp. $C(\mathcal{H}, w)$), as stated by the following corollary.

**Corollary 3.5** *Let* $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ *be an* EC-*structure and* $w \in W_{\mathcal{H}}$ *be a partial order. It holds that:*

$$\begin{array}{rcl} \Box MVI(\mathcal{H}, w) & = & MVI(\mathcal{H}, w) \cap S(\mathcal{H}, w); \\ \Diamond MVI(\mathcal{H}, w) & = & MVI(\mathcal{H}, w) \cup C(\mathcal{H}, w). \blacksquare \end{array}$$

In Section 6, we will exploit Corollary 3.5 to devise an algorithm for *MVI*s computation in *MEC*. Furthermore, from Corollary 3.5 it is immediate to conclude that the sets of necessary *MVI*s, *MVI*s, and possible *MVI*s with respect to the current state of knowledge form an inclusion chain as formally stated by the following proposition.

**Proposition 3.6** (*Necessary* MVI*s and possible* MVI*s enclose* MVI*s*)

*Let* $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ *be an* EC-*structure and* $w \in W_{\mathcal{H}}$ *be a partial order. It holds that*

$$\Box MVI(\mathcal{H}, w) \subseteq MVI(\mathcal{H}, w) \subseteq \Diamond MVI(\mathcal{H}, w). \qquad \blacksquare$$

Notice that if $w$ is a total order, then $S(\mathcal{H}, w) = \mathcal{L}(EC)$ and $C(\mathcal{H}, w) = \emptyset$, and thus $\Box MVI(\mathcal{H}, w) = \Diamond MVI(\mathcal{H}, w) = MVI(\mathcal{H}, w)$.

## 3.2. MVIs computation and updates

In this section we discuss the problem of determining how the acquisition of further information about the set of event occurrences and/or their occurrence times may affect the behaviour of *EC* and *MEC*. We first discuss updates of ordering information; then, we change the perspective and analyze the effects of acquiring new event occurrences.

Given an *EC*-structure $\mathcal{H}$, we want to study the behaviour of the sets of true, necessarily true and possibly true *MVI*s with respect to the acquisition of new ordering information [4]. When the arrival of a new piece of ordering information causes a transition into a more refined state of knowledge, the current set of *MVI*s may vary in two different ways. On the one hand, the update may create a new MVI by connecting an event $e_1$, initiating a property $p$, to an event $e_2$ terminating $p$. On the other hand, a new link can transform a previously innocuous event $e$ into an interrupting event for a current MVI $p(e_1, e_2)$. This allows us to conclude that the function $MVI(\mathcal{H}, \cdot)$ is nonmonotonic with respect to the evolution of the ordering information. On the contrary, $S(\mathcal{H}, \cdot)$ and $C(\mathcal{H}, \cdot)$ possess a monotonic behavior: the set $S(\mathcal{H}, \cdot)$ grows monotonically as the current ordering information is refined, while the set $C(\mathcal{H}, \cdot)$ shrinks monotonically.

However, even though $MVI(\mathcal{H}, \cdot)$ has a nonmonotonic behaviour, it is possible to show that its intersection (resp. union) with $S(\mathcal{H}, \cdot)$ (resp. $C(\mathcal{H}, \cdot)$) does not shrink (resp. grow) when the current partial order is updated with new consistent pairs of events. We first prove that for any pair $w, w' \in W$, with $w \subseteq w'$, $MVI(\mathcal{H}, w) \cap S(\mathcal{H}, w) \subseteq MVI(\mathcal{H}, w') \cap S(\mathcal{H}, w')$. To this end, it suffices to prove that if $p(e_1, e_2) \in MVI(\mathcal{H}, w) \setminus MVI(\mathcal{H}, w')$, then $p(e_1, e_2) \notin S(\mathcal{H}, w)$. From $p(e_1, e_2) \in MVI(\mathcal{H}, w)$ and $p(e_1, e_2) \notin MVI(\mathcal{H}, w')$, it follows that moving from $w$ to $w'$ transforms a previously innocuous event $e$ into an interrupting event for $p(e_1, e_2)$. This means that the event $e$ affects either $p$ or a property incompatible with $p$ and $e$ is located between $e_1$ and $e_2$ in $w'$, while it is unordered with respect to $e_1$ or $e_2$ in $w$. By the definition of $S(\mathcal{H}, \cdot)$, this allows us to conclude that $p(e_1, e_2) \notin S(\mathcal{H}, w)$. In a similar way, we can prove that $MVI(\mathcal{H}, w') \cup C(\mathcal{H}, w') \subseteq MVI(\mathcal{H}, w) \cup C(\mathcal{H}, w)$. To this end, it suffices to prove that if $p(e_1, e_2) \in MVI(\mathcal{H}, w') \setminus MVI(\mathcal{H}, w)$, then $p(e_1, e_2) \in C(\mathcal{H}, w)$. From $p(e_1, e_2) \in MVI(\mathcal{H}, w')$ and $p(e_1, e_2) \notin MVI(\mathcal{H}, w)$, it follows that moving from $w$ to $w'$ creates a new *MVI* $p(e_1, e_2)$ by connecting an event $e_1$, that initiates $p$, to an event $e_2$, that terminates $p$. This means that the events $e_1$ and $e_2$, that respectively initiate and terminate $p$, are ordered in $w'$ and unordered in $w$, and thus, by the definition of $C(\mathcal{H}, \cdot)$, $p(e_1, e_2) \in C(\mathcal{H}, w)$. Exploiting Corollary 3.5, this allows us to prove the following proposition.

**Proposition 3.7** (*Monotonicity of necessary and possible* MVI*s w.r.t. the addition of further ordering information*)

*Let* $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ *be an EC-structure and* $w, w' \in W_{\mathcal{H}}$ *be two partial orders. It holds that:*

*a. if* $w \subseteq w'$, *then* $\Box MVI(\mathcal{H}, w) \subseteq \Box MVI(\mathcal{H}, w')$;

*b. if* $w \subseteq w'$, *then* $\Diamond MVI(\mathcal{H}, w') \subseteq \Diamond MVI(\mathcal{H}, w)$. $\blacksquare$

By combining Propositions 3.6 and 3.7, we have that $\Box MVI(\mathcal{H}, \cdot)$ and $\Diamond MVI(\mathcal{H}, \cdot)$ constrain the variability of the set of *MVI*s derivable using *EC*. The state of minimum information corresponds to the absence of any ordering data: $\Box MVI(\mathcal{H}, \cdot)$ and $MVI(\mathcal{H}, \cdot)$ derive no formula, while $\Diamond MVI(\mathcal{H}, \cdot)$ derives all consistent property-labeled intervals. As new ordering information arrives, $\Box MVI(\mathcal{H}, \cdot)$ increases, $\Diamond MVI(\mathcal{H}, \cdot)$ decreases, but $MVI(\mathcal{H}, \cdot)$ always sits somewhere between them. When enough ordering information has been entered (at the limit, when the set of events has been completely ordered) $\Box MVI(\mathcal{H}, \cdot)$ and $\Diamond MVI(\mathcal{H}, \cdot)$ meet at a common value, constraining $MVI(\mathcal{H}, \cdot)$ to assume that same value.

We now consider the evolution of the sets of *MVI*s, necessarily true *MVI*s, and possibly true *MVI*s in the case in which the knowledge state $w$ remains unchanged and the *EC*-structure $\mathcal{H}$ is refined thanks to the acquisition of new event occurrences. Even though the addition of a new event occurrence always causes a transition into a richer *EC*-structure, the set of true *MVI*s remains stable, since no ordering information about the entered event occurrence is added. On the contrary, the set $S(\cdot, w)$ can only shrink as new events arrive, while the set $C(\cdot, w)$ grows monotonically. Taking advantage of Corollary 3.5, we can immediately prove the following proposition.

**Proposition 3.8** (*Monotonicity of* $\Box$*- and* $\Diamond$*-*MVI*s w.r.t. the addition of new event occurrences*)

*Let* $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ *and* $\mathcal{H}' = (E', P, [\cdot\rangle', \langle\cdot]', ]\cdot,\cdot[)$ *be two EC-structures, such that* $E \subseteq E'$ *and* $[\cdot\rangle'$ *and* $\langle\cdot]'$ *respectively extend* $[\cdot\rangle$ *and* $\langle\cdot]$ *to model the effects of the events in* $E' \setminus E$ *on the properties in* $P$*, and let* $w$ *be a partial order. It holds that:*

*a.* $\Box\Phi(\mathcal{H}', w) \subseteq \Box\Phi(\mathcal{H}, w)$*;*

*b.* $\Diamond\Phi(\mathcal{H}, w) \subseteq \Diamond\Phi(\mathcal{H}', w)$*.* ■

It is worth noting that, whenever we allow the addition of both ordering information and new event occurrences, it is impossible to identify any general rule constraining the behaviour of $MVI(\cdot, \cdot)$, $\Diamond MVI(\cdot, \cdot)$, and $\Box MVI(\cdot, \cdot)$.

**Example 3.9** (*Beverage dispenser*)

We illustrate the relationships between *MVI*s computation in *EC* and *MEC* and updates by means of a simple example. We want to model the operations of a simple beverage dispenser [4]. It can output either apple juice or orange juice (but not both simultaneously). The choice is made by means of a selector with three positions (*apple*, *orange* and *stop*): by setting the selector to the *apple* or to the *orange* position, apple juice or orange juice is obtained, respectively; choosing the *stop* position terminates the production of juice. In our example, we distinguish three types of events corresponding to the various settings of the selector and two relevant properties, *supplyApple* and *supplyOrange*, indicating that apple juice or orange juice is being dispensed, respectively. The event of setting the selector to the *apple* (*orange*) position *initiates* the property *supplyApple* (*supplyOrange*), while setting it to the *stop* position *terminates* both properties. The properties *supplyApple* and *supplyOrange* are *exclusive* since apple juice and orange juice cannot be output simultaneously.

| *Update* | EC | SKEC | CREC |
|---|---|---|---|
| | $a(e_1, e_4)$ | $a(e_1, e_4)$ | $a(e_1, e_4)$ |
| $e_3$ | $a(e_1, e_4)$ | Ø | $a(e_1, e_4)$, $o(e_3, e_4)$ |
| $e_2$ | $a(e_1, e_4)$ | Ø | $a(e_1, e_2)$, $a(e_1, e_4)$, $o(e_3, e_2)$, $o(e_3, e_4)$ |
| $(e_1, e_2)$ | $a(e_1, e_4)$ | Ø | $a(e_1, e_2)$, $a(e_1, e_4)$, $o(e_3, e_2)$, $o(e_3, e_4)$ |
| $(e_2, e_4)$ | $a(e_1, e_2)$ | Ø | $a(e_1, e_2)$, $o(e_3, e_2)$ $o(e_3, e_4)$ |
| $(e_2, e_3)$ | $a(e_1, e_2)$ | $a(e_1, e_2)$ | $a(e_1, e_2)$, $o(e_3, e_4)$ |
| $(e_3, e_4)$ | $a(e_1, e_2)$, $o(e_3, e_4)$ | $a(e_1, e_2)$, $o(e_3, e_4)$ | $a(e_1, e_2)$, $o(e_3, e_4)$ |

**Figure 3.1. The Beverage Dispenser Example.**

We consider a scenario consisting of an event $e_1$, that initiates the property *supplyApple*, and a *stop* event $e_4$, that terminates both *supplyApple* and *supplyOrange*. Furthermore, we assume that $e_1$ precedes $e_4$. This scenario can be formalized as follows.

$$
\begin{aligned}
&E = \{e_1, e_4\};\\
&P = \{supplyApple, supplyOrange\};\\
&[supplyApple\rangle = \{e_1\};\\
&\langle supplyApple] = \langle supplyOrange] = \{e_4\};\\
&]supplyApple, supplyOrange[.
\end{aligned}
$$

We describe the evolution of the sets of true *MVI*s, necessarily true *MVI*s, and possibly true *MVI*s when the following sequence of database updates is performed: (i) an event $e_3$, that initiates the property *supplyOrange*, is added; (ii) an event $e_2$, that terminates both properties, is inserted; (iii) the following sequence of ordered pairs of events is entered: $(e_1, e_2)$, $(e_2, e_4)$, $(e_2, e_3)$, and $(e_3, e_4)$. In Figure 3.1, we describe the effects of each update on these three sets. The first row of the table reports their initial values; each subsequent row is associated with an update to the database and filled in with the corresponding values of the three sets. The first column shows the performed update; the second column contains the *MVI*s derived by *EC*, where $a(e_i, e_t)$ (resp. $o(e_i, e_t)$) is a shorthand for the statement that property *supplyApple* (resp. *supplyOrange*) holds between $e_i$ and $e_t$; the third and fourth columns contain the sets of necessary and possible *MVI*s, respectively.

The fact that, when the pair $(e_2, e_4)$ is entered, the *MVI* $a(e_1, e_4)$ disappears, while a new *MVI* $a(e_1, e_2)$ is added, provides an example of the nonmonotonic behavior of $MVI(\cdot, \cdot)$ with respect to the addition of ordering information. As for the monotonic behavior of $\Box MVI(\cdot, \cdot)$ and $\Diamond MVI(\cdot, \cdot)$, we can observe that the set of necessary *MVI*s grows (resp. shrinks) when new ordered pairs of events (resp. event occurrences) are acquired, while the set of possibly *MVI*s shrinks (resp. grows) as new ordering information (resp. information about event occurrences) is added. Finally, observe that the set of *MVI*s always lies somewhere between $\Box MVI(\cdot, \cdot)$ and $\Diamond MVI(\cdot, \cdot)$, and, when the ordering information is complete, the three sets meet at a common value.

## 3.3. Existing algorithms for MVIs computation

Given an EC-structure $\mathcal{H}$ and a knowledge state $w$, the set of *MVI*s for a given property $p$, with respect to $\mathcal{H}$ and $w$, can be computed according to two alternative temporal reasoning strategies: a generate-and-test strategy and a generate-only one. The *generate-and-test* strategy first generates all ordered pairs of initiating and terminating events for $p$, and then, for every pair, it verifies whether there are known interrupting events in between or not. On the contrary, the *generate-only* strategy identifies possible interrupting events during the search of candidate *MVI*s for $p$, i.e. pairs $(e_1, e_2)$ such that $e_1$ initiates $p$ and $e_2$ terminates $p$. Generate-only strategies generally leads to the development of algorithms for *MVI*s computation with a lower worst-case complexity.

Traditional algorithms for *MVI*s computations adopt the simpler *generate-and-test* strategy, which can be easily derived from the specification of *EC* semantics given in Definition 3.2 [5, 11] (this strategy can be easily adapted to *MEC* by exploiting the local conditions given in Proposition 3.4 [2, 11]). In order to compute all *MVI*s $p(e_1, e_2)$, with respect to $w$, such algorithms first generate all consistent intervals $(e_1, e_2)$ such that $e_1$ initiates $p$, $e_2$ terminates $p$, and $e_1 <_w e_2$; then, they check whether or not the validity of $p$ is broken during the interval $(e_1, e_2)$. Such algorithms can be easily proved to be sound and complete with respect to the semantics of *EC*, but they are quite expensive: they operate in $\mathcal{O}(n^5)$ time, where $n$ is the number of events [3, 6].

A generate-only algorithm for *MVI*s computation can be found in [6]. Such an algorithm operates on the transitive reduction of the given partial ordering, which needs to be updated (paying a non-constant cost) whenever further ordering information is entered in the database. The behavior of this algorithm can be described as follows: for any given property $p$ and any event $e_1$ initiating $p$, the algorithm examines all events accessible from $e_1$, searching for events terminating $p$. The search starts from the successors of $e_1$, and proceeds breadth-first. The nodes which are directly accessible from $e_1$ (nodes that belong to the first layer) can be partitioned into two categories: interrupting events, that is, events that affect either $p$ or a property incompatible with $p$, and independent events, that is, events that affect neither $p$ nor properties incompatible with $p$. Events belonging to the first category, which terminate $p$, contribute to the set of *MVI*s for $p$ initiated by $e_1$, and are returned to the user; moreover, nodes which are reachable from them are marked, since there is no need to keep them into consideration during further processing. The remaining nodes belonging to the first category are marked, together with their direct and indirect successors, because they cannot belong to a successful path for the user query. Nodes belonging to the second category are used to determine the next layer to explore, which consists of the collection of all their unmarked successors. The procedure repeats recursively these steps until the last layer is reached.

It is possible to show that this strategy is sound and complete whenever every property is incompatible with all the other ones. In particular, it is sound and complete whenever the set of properties $P$ is a singleton set (the single-property case studied in [6]). In such a restricted context, *MVI*s computation can actually be simplified. Whenever all event occurrences affect the same property $p$, any interval $(e_1, e_2)$ is an *MVI* for $p$ if and only if $e_1$ initiates $p$, $e_2$ terminates $p$, and $e_1$ is an immediate predecessor of $e_2$, that is, there are no recorded events in between, with respect to the transitive reduction of the given partial ordering. Unfortunately, in the general case, in spite of the conjecture formulated in [6], such an algorithm is complete, but not sound. A simple counterexample will be provided in Section 4. In the next section, we propose an efficient, sound and complete generate-only algorithm for the *MVI*s computation problem, which successfully pairs breadth-first and depth-first visits of the graph representing the given partial order of events.

## 4. A sound and complete generate-only algorithm for *MVI*s computation

In this section, we describe a new generate-only algorithm that computes the set of *MVI*s which are true with respect to a partial order $w$ and an *EC*-structure $\mathcal{H}$. We provide a high-level description of the algorithm and prove its soundness and completeness with respect to the semantics of *EC*.

Let $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ be an EC-structure and $o \in O_E$ (denoted by $O_{\mathcal{H}}$ hereinafter) be an acyclic binary relation. We define an algorithm for *MVI*s computation that combines a breadth-first and a depth-first visit of the graph

$(E, o)$, which is directed and acyclic, but not necessarily connected (background knowledge on elementary graph algorithms can be found in [8]). In the following, whenever it does not lead to ambiguities, we denote the graph $(E, o)$ by $G$ and the subgraph of $(E, o)$ generated by $e$ by $G(e)$.

The algorithm behaves as follows: for every property $p \in P$ and every event $e_1 \in E$ initiating $p$, it searches the graph $G(e_1)$ for all events $e_2$ such that the interval $(e_1, e_2)$ is an *MVI* for $p$. Given a property $p$ and an event $e_1$, the algorithm associates the following labels with the nodes of $G(e_1)$:

- `unmarked`: it denotes nodes (events) to be visited;
- `visited`: it denotes nodes (events) already visited;
- `marked`: it denotes nodes (events) that initiate or terminate either $p$ or a property incompatible with $p$;
- `cutoff`: it denotes nodes (events) which are cut off from the search space, because they cannot terminate any *MVI* for $p$ initiated by $e_1$.

The set of events $e_2$ such that $p(e_1, e_2)$ is an *MVI* is computed as follows. Initially, all nodes in $G(e_1)$ are labeled with `unmarked`; then, the graph $G(e_1)$ is visited breadth-first. The breadth-first visit of $G(e_1)$ starts from the successors of $e_1$ (first layer) and proceeds, layer by layer, until the last layer is reached. The last layer is a layer followed by an empty layer; since $G(e_1)$ is acyclic, such a layer always exists and it is unique. At each layer, only `unmarked` events are processed. Let $e$ be an `unmarked` event belonging to the current layer. The algorithm labels $e$ as `visited` and checks whether or not it initiates or terminates either $p$ or a property incompatible with $p$. If the outcome of the test is positive, then the following operations are executed before processing the next event of the layer:

1. $e$ is labeled as `marked`;
2. the label `cutoff` is assigned to all nodes of $G(e)$ different from $e$;
3. if $e$ terminates $p$, then the node $e$ is saved.

Once the whole graph $G(e_1)$ has been visited, all the saved nodes, which are still labeled as `marked`, are returned; they are all and only the events that terminate an *MVI* for $p$ initiated by $e_1$.

A pseudo-code description of such an algorithm for *MVI*s computation can be given as follows.

```
$MVI \leftarrow \emptyset$
**for each** $p \in P$ **do**
    **for each** $e_1 \in [p\rangle$ **do**
        $S \leftarrow \emptyset$
        **for each** $e \in G(e_1)$ **do**
            set($e$, unmarked)
        L ← nextlayer($\{e_1\}$)
        **while** $L \neq \emptyset$ **do**
            **for each** $e \in L$ **do**
                **if** $is\_relevant\_to$($e$, $p$) **then**
                    set($e$, marked)
                    cutoff($e$)
                    **if** $e \in \langle p]$ **then**
                        $S \leftarrow S \cup \{e\}$
            L ← nextlayer(L)
        **for each** $e_2 \in S$ **do**
            **if** label($e_2$, marked) **then**
                $MVI \leftarrow MVI \cup \{p(e_1, e_2)\}$
**return** $MVI$
```

The procedure `set(e,l)` assigns the label `l` to the event `e`, the boolean function `label(e,l)` checks whether the label `l` is associated with the event `e` or not, and the boolean function `is_relevant_to(e,p)` tests whether or not `e` initiates or terminates either `p` or a property incompatible with `p`. The procedure `nextlayer(L)` computes the next layer in the breadth-first visit of the graph $G(e_1)$:

```
nextlayer(L)
    $L' \leftarrow \emptyset$
    **for each** $e \in L$ **do**
        **if** label($e$, unmarked) **or** label($e$, visited) **then**
            **for each** successor $e'$ of $e$ **do**
                **if** label($e'$, unmarked) **then**
                    set($e'$, visited)
                    $L' \leftarrow L' \cup \{e'\}$
    **return** L'
```

Finally, the procedure `cutoff(l)` visits depth-first the subgraph generated by the event `e` and labels as `cutoff` all its nodes:

```
cutoff(e)
    **for each** successor $e'$ of $e$ **do**
        **if not** label($e'$, cutoff) **then**
            set($e'$, cutoff)
            cutoff($e'$)
```

Before proving that such an algorithm in sound and complete with respect to the semantics of *EC*, we illustrate its behaviour by means of two simple examples. Let $e_1$, $e_2$, and $e_3$ be three event occurrences, $p$ and $q$ be two incompatible properties, and $o = \{(e_1, e_2), (e_1, e_3), (e_2, e_3)\}$ be the current knowledge state (cf. Figure 4.2, left side). Suppose that $e_1$ initiates $p$, $e_2$ initiates $q$, and $e_3$ terminates $p$. The set of *MVI*s for $p$, which are initiated by $e_1$, is computed as follows. The algorithm first labels as `unmarked` all nodes of $G(e_1)$, and then it visits breadth-first $G(e_1)$. The first layer contains both $e_2$ and $e_3$. Suppose that the algorithm first processes $e_3$ and then $e_2$. The node $e_3$ is labeled as `marked` and saved, because it terminates $p$. The

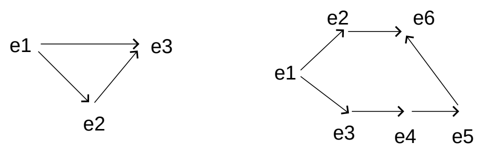

**Figure 4.2. Two graphs representing ordering relations**

propagation of the label `cutoff` has no effect, since $e_3$ has no successors. Hence, the node $e_2$ is processed and labeled as `marked`, because it initiates a property $q$ which is incompatible with $p$. The effect of propagating the label `cutoff` is that of replacing the label `marked` of $e_3$ by the label `cutoff`. Then, the visit of $G(e_1)$ terminates (all nodes have already been visited) and the algorithm returns no *MVI*s for $p$ initiated by $e_1$, because the label associated with $e_2$ (the only saved event) is `cutoff` and not `marked`. This example should clarify the role of the label `cutoff`: some events may be labelled as `marked` along a "short" path ($e_1 \rightarrow e_3$ in the example) and saved as candidate ending points of an *MVI* for the considered property. However, an interval is an *MVI* for a property if and only if *all* paths leading from the initiating event to the terminating one do not contain interrupting events, that is, events that initiate or terminate either the considered property or a property incompatible with it. If there exists a longer path ($e_1 \rightarrow e_2 \rightarrow e_3$ in the example) which contains an interrupting event, then the candidate node is cut off during the propagation of the label `cutoff`.

It is worth noticing that the event graph of the example contains a transitive edge ($e_1 \rightarrow e_3$). The next example shows that `cutoff` labels are needed also for reasoning about event graphs devoid of transitive edges (their use can be avoided only if we restricted ourselves to multiple incompatible properties). Consider a scenario consisting of six event occurrences $e_1, e_2, e_3, e_4, e_5$, and $e_6$, two incompatible properties $p$ and $q$, and the knowledge state $o$ depicted in Figure 4.2, right side, which has no transitive edges. Suppose that $e_1$ initiates $p$, $e_5$ initiates $q$, $e_6$ terminates $p$, and $e_2$, $e_3$, and $e_4$ affect neither $p$ nor a property incompatible with $p$. The interval $(e_1, e_6)$ is not an *MVI* for $p$, because there exists an interrupting event, namely $e_5$, which occurs between $e_1$ and $e_6$. The algorithm removes the node $e_6$ from the set of candidate terminating events associated with initiating event $e_1$ when it propagates the label `cutoff` during the processing of $e_5$.

The following theorem proves that the proposed algorithm computes exactly the set of *MVI*s as defined in Definition 3.2.

**Theorem 4.1** *The proposed generate-only algorithm is sound and complete.*

**Proof.**

We first prove that the algorithm is sound, that is, if $p(e_1, e_2)$ is generated by the algorithm, then $p(e_1, e_2)$ is an *MVI*. Given a property $p$ and an event $e_1$, the algorithm searches the acyclic graph $G(e_1)$ for terminating events $e_2$. Since the visit is breadth-first, each node is reached along the shortest path on $G(e_1)$ starting from $e_1$. Given a node $e$, we denote by $D(e)$ the length of the shortest path on $G(e_1)$ connecting $e_1$ to $e$. We show that $p(e_1, e_2)$ is an *MVI* if and only if $e_1$ initiates $p$, $e_2$ terminates $p$, $e_2$ belongs to $G(e_1)$, and every path $e_1 \rightsquigarrow e_2$ from $e_1$ to $e_2$ in $G(e_1)$ does not contain interrupting events for $p$, that is, events that affect either $p$ or a property incompatible with $p$ and differ from both $e_1$ and $e_2$.

We proceed by contradiction. Suppose that $p(e_1, e_2)$ is returned by the algorithm, but it is not an *MVI*. If $e_1$ does not initiate $p$ or $e_2$ does not terminate $p$, then $p(e_1, e_2)$ cannot be retrieved. Moreover, if $e_2$ does not belong to $G(e_1)$, then the visit of $G(e_1)$ does not retrieve $e_2$, and hence $p(e_1, e_2)$ cannot be generated. Finally, suppose that there exists at least one path $e_1 \rightsquigarrow e_2$ in $G(e_1)$ that contains at least one node $z$ which affects either $p$ or a property incompatible with $p$ and is different from $e_1$ and $e_2$. If $D(z) < D(e_2)$, then the node $z$ is visited before $e_1$, it is labeled as `marked`, and the label `cutoff` is propagated to the nodes of $G(z)$ different from $z$. In particular, $e_2$ is labeled as `cutoff` during such a propagation and thus it cannot be chosen as the terminating event of an *MVI* for $p$ initiated by $e_1$. Hence, $p(e_1, e_2)$ cannot be generated by the algorithm. If $D(z) > D(e_2)$ (notice that $D(z) \neq D(e_2)$, since $z \neq e_2$ and there are not simultaneous events), then the node $e_2$ is visited before $z$, it is labeled as `marked`, and the label `cutoff` is propagated to the nodes of $G(e_2)$ different from $e_2$. Since the graph $G(e_1)$ is acyclic and there exists a path from $z$ to $e_2$, there are no paths from $e_2$ to $z$; hence the propagation of the label `cutoff` does not reach the node $z$. The node $z$ is processed at some later stage, it is labeled as `marked`, and the label `cutoff` is propagated to the nodes of $G(z)$ different from $z$. In particular, the label of $e_2$ is changed from `marked` to `cutoff`, and thus $p(e_1, e_2)$ cannot be generated by the algorithm.

We now prove that the algorithm is complete, that is, if $p(e_1, e_2)$ in an *MVI*, then $p(e_1, e_2)$ is generated by the algorithm. Since $(e_1, e_2)$ is an interval, $e_2$ is reachable from $e_1$ in the graph $G(e_1)$. Since $p(e_1, e_2)$ is an *MVI*, every path $e_1 \rightsquigarrow e_2$ from $e_1$ to $e_2$ in $G(e_1)$ does not contain interrupting events for $p$ different from $e_1$ and $e_2$. Hence, the node $e_2$ is not cut off and, since it terminates $p$, it is labeled as `marked` and retrieved as the terminating event of an *MVI* for $p$ initiated by $e_1$. Thus, $p(e_1, e_2)$ is generated by the algorithm. ■

The proposed strategy is a *forward* strategy: given a property $p$ and an initiating event $e_1$, it visits the graph $G(e_1)$, looking for a terminating event $e_2$ such that $p(e_1, e_2)$ is an *MVI*. Nothing prevents us to define an equivalent backward strategy as follows. Given a directed graph $G$, let us denote by $\hat{G}$ the graph in which each edge $(e_i, e_j)$ has been replaced by the edge $(e_j, e_i)$. Given a property $p$ and a terminating event $e_2$, we visit the graph $\hat{G}(e_2)$ as before, looking for initiating events $e_1$ such that $p(e_1, e_2)$ is an *MVI*.

## 5 Complexity analysis

In this section, we analyze the worst-case computational complexity of the proposed algorithm for *MVI*s derivation.

Given an EC-structure $\mathcal{H}$ and an acyclic binary relation $o \in O_{\mathcal{H}}$, we determine the complexity of computing the set of *MVI*s with respect to $o$ and $\mathcal{H}$, i.e. the set of formulas $p(e_1, e_2)$ such that $o^+ \models p(e_1, e_2)$, by means of the proposed generate-only algorithm. We measure the complexity in terms of the size $n$ of the structure $\mathcal{H}$ (where $n$ is the number of recorded events) and the size $m$ of the relation $o$. Given an EC-structure $\mathcal{H}$, the set $E$ of events can be arbitrarily large, while the set $P$ of properties is fixed once and for all, since it is an invariant characteristic of the considered domain. Since the cardinality of $P$ does not change from one problem instance to another one (unless we change the application domain), while the cardinality of $E$ may grow arbitrarily, we choose the cardinality of $E$, that is, the number $n$ of events, as the size of $\mathcal{H}$ and consider the number of properties as a constant. Furthermore, we assume that verifying the truth of the propositions "*e initiates p*" and "*e terminates p*" costs $\mathcal{O}(1)$. Since the number of properties is constant, the tests "*e affects either p or a property incompatible with p*" and "*p is incompatible with q*" cost $\mathcal{O}(1)$ too.

The parameters $n$ and $m$ are equal to the number of nodes and the number of edges of the graph $(E, o)$ which is visited during the computation, respectively. Moreover, we have that $m = \mathcal{O}(n^2)$ when the event graph is dense, $m = \mathcal{O}(n)$ for sparse event graphs, and $m = \mathcal{O}(1)$ when only a constant number of events is ordered in $o$.

The following theorem proves that, under the above assumptions, the complexity of the proposed algorithm is quadratic for sparse event graphs and cubic for dense ones.

**Theorem 5.1** *The complexity of the generate-only algorithm is $\mathcal{O}(n \cdot m)$.*

**Proof.**

For every property $p$ and every event $e_1$ initiating $p$, the algorithm visits the graph $G(e_1)$ and retrieve all the events $e_2$ such that $p(e_1, e_2)$ is an *MVI*. Since the number of properties is constant, the complexity is $\mathcal{O}(n \cdot f(n, m))$, where $f(n, m)$ is the complexity of the procedure that visits the graph $G(e_1)$ and retrieves the nodes that terminate the *MVI* for $p$ initiated by $e_1$. It holds that $f(n, m)$ is the sum of the costs of the visit of $G(e_1)$ and of the processing of the nodes of $G(e_1)$.

The graph $G(e_1)$ is visited breadth-first to construct the layers and to retrieve the terminating events, while it is visited depth-first to propagate the labels `cutoff`. Each edge of the graph $G(e_1)$ is visited at least once (depth-first or breadth-first) and at most twice (first breadth-first, and then depth-first). Indeed, if an edge $(e_1, e_2)$ is depth-first visited, then $e_1$ is labeled as `marked` or `cutoff`. Hence, neither a breadth-first visit nor a depth-first one will later reconsider it. However, edges which have been already breadth-first visited can also be visited depth-first in order to propagate the label `cutoff`. It follows that the cost of visiting $G(e_1)$ is $\mathcal{O}(m)$.

Similarly, each node of the graph $G(e_1)$ is processed at least once (depth-first or breadth-first) and at most twice (first breadth-first, and then depth-first). Indeed, if the depth-first visit cuts off a node, then it will not be processed anymore. However, nodes labeled as `marked` or `visited`, which have been already processed during the breadth-first visit, can also be processed during a depth-first visit and labeled as `cutoff`. The processing of a node consists of the operations of labeling and testing for interrupting or terminating events. Both these operations cost $\mathcal{O}(1)$. Therefore, processing all nodes of $G(e_1)$ costs $\mathcal{O}(n)$.

Putting together the results of our analysis, we can conclude that $f(n, m) = \mathcal{O}(m) + \mathcal{O}(n) = \mathcal{O}(m + n)$. If $m = \mathcal{O}(1)$, then only a constant number of nodes is processed, and hence $f(n, m) = \mathcal{O}(1)$; otherwise, $n = \mathcal{O}(m)$, and thus $f(n, m) = \mathcal{O}(m)$. This allows us to conclude that the cost of the algorithm is $\mathcal{O}(n \cdot f(n, m)) = \mathcal{O}(n \cdot m)$. In particular, if the event graph is dense, that is, $m = \mathcal{O}(n^2)$, then the complexity is $\mathcal{O}(n^3)$, while if it is sparse, that is, $m = \mathcal{O}(n)$, then the cost is $\mathcal{O}(n^2)$. ■

## 6. The generalization to MEC

Given an EC-structure $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot], ]\cdot,\cdot[)$ and a partial order $w$, two efficient algorithms, that respectively compute necessary and possible *MVI*s with respect to $\mathcal{H}$ and $w$, can be obtained from Corollary 3.5 taking advantage of the algorithm for *MVI*s computation in *EC*. In order to compute the sets $C(\mathcal{H}, w)$ and $S(\mathcal{H}, w)$ (cf. Section 3), we proceed as follows. The elements of $C(\mathcal{H}, w)$ are obtained by selecting all property-labeled pairs of events $p(e', e'')$ such that $e'$ initiates $p$, $e''$ terminates $p$, and $e'$ and $e''$ are unordered in $w$:

```
$C \leftarrow \emptyset$
**for each** $p \in P$ **do**
    **for each** $(e_1, e_2) \in E \times E$ **do**
        **if** $e_1 \in [p\rangle$ **and** $e_2 \in \langle p]$ **and**
          $(e_1, e_2) \notin w$ **and** $(e_2, e_1) \notin w$ **then**
          $C \leftarrow C \cup \{p(e_1, e_2)\}$
**return** $C$
```

The computation of $S(\mathcal{H}, w)$ is more involved. First, we compute the set $U(\mathcal{H}, w)$, containing all pairs $(e, p) \in E \times P$ such that there exists another event $e'$, which affects either $p$ or a property incompatible with $p$ and is unordered with respect to $e$ in $w$. It is easy to see that if $(e, p) \in U(\mathcal{H}, w)$, then $e$ neither initiates nor terminates a $\Box$-*MVI* for $p$. The set $S(\mathcal{H}, w)$ is obtained by selecting those atomic formulas $p(e_1, e_2)$ such that $e_1$ initiates $p$, $e_2$ terminates $p$, and neither $(e_1, p)$ nor $(e_2, p)$ belong to $U(\mathcal{H}, w)$:

```
// *compute* $U(\mathcal{H}, w)$
$U \leftarrow \emptyset$
$S \leftarrow \emptyset$
**for each** $p \in P$ **do**
    **for each** $e \in E$ **do**
        Found $\leftarrow$ False
        $V \leftarrow E$
        **while not** Found **and** $V \neq \emptyset$ **do**
              let $e' \in V$
              **if** $(e, e') \notin w$ **and**
                $(e', e) \notin w$ **and**
                $e'$ $is\_relevant\_to(e', p)$ **then**
                Found $\leftarrow$ True
                $U \leftarrow U \cup \{(e, p)\}$
              **else**
                $V \leftarrow V \setminus \{e'\}$
// *compute* $S(\mathcal{H}, w)$ *taking advantage of* $U(\mathcal{H}, w)$
**for each** $p \in P$ **do**
    **for each** $(e_1, e_2) \in E \times E$ **do**
        **if** $(e_1, p) \notin U$ **and**
          $(e_2, p) \notin U$ **then**
          $S \leftarrow S \cup \{p(e_1, e_2)\}$
**return** $S$
```

In order to determine the set of necessarily true *MVI*s, it suffices to compute the sets $MVI(\mathcal{H}, w)$ (as proposed in Section 4) and $S(\mathcal{H}, w)$ (as explained above); the set $\Box$MVI$(w)$ can be obtained by intersecting them. Similarly, possibly true *MVI*s are obtained taking the union of $MVI(\mathcal{H}, w)$ and $C(\mathcal{H}, w)$. The proof of soundness and completeness easily follows from Corollary 3.5 and Theorem 4.1.

**Theorem 6.1** *The proposed algorithms for necessary and possible* MVI*s computation are sound and complete.* ■

The following theorem states that the complexity of the algorithms for necessary and possible *MVI*s computation is (slightly) higher than that of the algorithm for basic *MVI*s only in the case of sparse event graphs.

**Theorem 6.2** *The complexity of the algorithms for necessary and possible* MVI*s computation is* $\mathcal{O}(n \cdot m + n^2 \cdot \log n)$.

**Proof.**

Given a knowledge state $w$, the algorithm for the computation of $MVI(\mathcal{H}, w)$ has complexity $\mathcal{O}(n \cdot m)$ (Theorem 5). Moreover, it is immediate to see that determining the sets $C(\mathcal{H}, w)$ and $S(\mathcal{H}, w)$ costs $\mathcal{O}(n^2)$. Finally, taking the intersection (resp. union) of two sets of cardinality $r$ costs $\mathcal{O}(r \cdot \log r)$. Since $MVI(\mathcal{H}, w)$, $C(\mathcal{H}, w)$, and $S(\mathcal{H}, w)$ have cardinality $\mathcal{O}(n^2)$, the overall cost is $\mathcal{O}(n \cdot m + n^2 \cdot \log n)$. ■

## 7. Conclusions and further developments

In this paper, we outlined a graph-theoretic approach to the problem of efficiently reasoning about partially ordered events in Kowalski and Sergot's Event Calculus [11]. The proposed algorithm exploits a generate-only strategy based on a graph representation of ordering information that reduces the computation of the *MVI*s to a visit of the event graph that pairs traditional breadth-first and depth-first searches. Furthermore, we showed how the proposed strategy can be extended to deal with the Modal Event Calculus [4].

In [6], Chittaro et alii propose a generate-only algorithm for *MVI*s computation that operates on the transitive reduction of the given partial ordering. Such an algorithm is sound and complete whenever every property is incompatible with all the other ones. In particular, it is sound and complete whenever there is only one property (single-property case). We are currently working at the development of a sound and complete algorithm that generalizes to the multi-property case the strategy discussed in [6]. The basic steps of this generalized strategy are the following ones: first, it computes (and maintains) the transitive closure $G^+ = \langle E, o^+\rangle$ of the graph $G$ representing the available ordering information; then, for every property $p$, it extracts from $G^+$ the subgraph induced by the set of events that initiate or terminate $p$, or a property incompatible with $p$; finally, it derives the set of *MVI*s for any property $p$ by applying the strategy for the single-property case to the transitive reduction of the subgraph for $p$. We expect to achieve complexity results comparable with the ones we reported in the present work.

## Acknowledgements

We would like to thank the anonymous reviewers for their useful comments. Thanks also to Roberto Fracas whose *Tesi di Laurea* contributed to the achievement of the results reported in this paper [10].

## References

[1] A. V. Aho, M. R. Garey, and J. Ullman. The transitive reduction of a directed graph. *SIAM Journal of Computing*, 1(2):131–137, 1972.

[2] I. Cervesato, L. Chittaro, and A. Montanari. A modal calculus of partially ordered events in a logic programming framework. In L. Sterling, editor, *Proceedings of the Twelfth International Conference on Logic Programming — ICLP'95*, pages 299–313, Kanagawa, Japan, 13–16 June 1995. MIT Press.

[3] I. Cervesato, M. Franceschet, and A. Montanari. A hierarchy of modal event calculi: Expressiveness and complexity. In H. Barringer, M. Fisher, D. Gabbay, , and G. Gough, editors, *Proceedings of the Second International Conference on Temporal Logic — ICTL'97*, pages 1–17, Manchester, England, 14–18 July 1997. Kluwer Applied Logic Series. To appear.

[4] I. Cervesato and A. Montanari. A general modal framework for the event calculus and its skeptical and credulous variants. *Journal of Logic Programming*, 38(2):111–164, 1999.

[5] I. Cervesato, A. Montanari, and A. Provetti. On the nonmonotonic behavior of the event calculus for deriving maximal time intervals. *International Journal on Interval Computations*, 2:83–119, 1993.

[6] L. Chittaro, A. Montanari, and I. Cervesato. Speeding up temporal reasoning by exploiting the notion of kernel of an ordering relation. In S. Goodwin and H. Hamilton, editors, *Proceedings of the Second International Workshop on Temporal Representation and Reasoning — TIME'95*, pages 73–80, Melbourne Beach, FL, 26 April 1995.

[7] L. Chittaro, A. Montanari, and A. Provetti. Skeptical and credulous event calculi for supporting modal queries. In A. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence — ECAI'94*, pages 361–365. John Wiley & Sons, 1994.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. The MIT Press, 1990.

[9] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36:375–399, 1988.

[10] R. Fracas. Uso di algoritmi su grafi per ragionare in modo efficiente su insiemi di eventi parzialmente ordinati (in Italian). Tesi di Laurea in Scienze dell'Informazione, Università di Udine, Italy, 1997.

[11] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[12] J. van Leeuwen. Graph algorithms. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity*, pages 525–632. Elsevier, 1990.