Pairing Transitive Closure and Reduction to Efficiently Reason about Partially Ordered Events

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 show how well-known graph-theoretic techniques can be successfully exploited to efficiently reason about partially ordered events in Kowalski and Sergot's Event Calculus and in its skeptical and credulous modal variants. We replace the traditional generate-and-test strategy of basic Event Calculus by a generate-only strategy that operates on the transitive closure and reduction of the underlying directed acyclic graph of events. We prove the soundness and completeness of the proposed strategy, and thoroughly analyze its computational complexity.

1 Introduction

Reasoning about actions and change is emerging as a major requirement in many intelligent system applications, including diagnosis, robotics, agent modelling, qualitative physics, monitoring, planning and plan validation, and natural language understanding. Considerable research efforts have been devoted to providing reasoning about actions and change tasks with a common formal basis, so that they can defined in a uniform framework. In contrast, efficiency issues have received a very limited attention, and this has been recognized as a major limiting factor to its application in realistic domains.

In this paper, we show how well-known graph-theoretic techniques can be successfully exploited to efficiently reason about partially ordered events in Kowalski and Sergot's Event Calculus (*EC* for short) [5] and in its skeptical and credulous modal variants [1]. Given a set of events, *EC* is able to infer the set of maximal validity intervals (*MVI*s) over which the properties initiated and/or terminated by them hold uninterruptedly. Events can be temporally qualified in several ways. We consider the relevant case where either the occurrence time of an event is totally unspecified or its relative temporal position with respect to (some of) the other events is given [3]. 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 and/or of further ordering information about the given events. The set of *MVI*s for any given property p has been traditionally computed at query time according to a simple (and expensive) *generate-and-test* strategy: *EC* first blindly picks up every candidate pair of events (e_i, e_t), where e_i and e_t respectively initiate and terminate p; then, it checks whether or not e_i precedes e_t; finally, it looks for possible events e that occur between e_i and e_t and interrupt the validity of p. Checking whether e_i precedes e_t or not reduces to establish if the edge (e_i, e_t) belongs to the transitive closure o^+ of o as well as checking if there exists an interrupting event e requires to verify if both (e_i, e) and (e, e_t) belong to o^+.

In [2], Chittaro et al. outline an alternative (and efficient) *generate-only* strategy for *MVI*s computation when all recorded events are concerned with the same unique property p. According to such a strategy, the graph $G = \langle E, o \rangle$ is replaced by its *transitive reduction* $G^- = \langle E, o^- \rangle$, which must be maintained whenever a new consistent and non-redundant pair of events (e_i, e_j) is entered (the addition of a new event e to E does not affect o^-). Since any event $e \in E$ either initiates or terminates p, the set of *MVI*s for p can be obtained by searching G^- for edges (e_i, e_j) such that e_i initiates p and e_j terminates it. Being G^- the transitive reduction of G ensures us that there are no interrupting events for p that occur between e_i and e_j. In this paper, we show how such a generate-only strategy can be generalized to the (general) multiple-property case of *EC*[1].

The paper is organized as follows. In Section 2, we introduce some background knowledge. In Section 3, we first revise the generate-only strategy for *MVI*s computation in *EC* in the single-property case, and, then, we generalize it to the multiple-property one. The increase in efficiency of the proposed strategy with respect to the traditional generate-and-test one is demonstrated by the complexity analysis of Section 4. In the last part of the paper, we briefly discuss the achieved results.

2 Preliminaries

In this section, we first introduce some basic notions about ordering relations, transitive closure and transitive reduction [7]; then, we briefly recall syntax and semantics of the basic Event Calculus [5] and of the Modal Event Calculus [1].

2.1 On ordering relations, transitive closure and reduction

Let us first remind some basic notions about ordering relations and ordered sets upon which we will rely in the following. *EC* usually *represents* ordering information as a binary acyclic relation o on the set of events E, that is, as

[1] The generalization to the multiple-property case of *EC* sketched in [2] 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.

an ordering relation possibly missing some transitive links, but it uses ordering information as a (strict) partial order w that can be recovered as the transitive closure o^+ of o.

Definition 1. (DAGs, 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.

In this paper, we will make a massive use of the notions of transitive closure and reduction of a *DAG*.

Definition 2. (Transitive closure and reduction 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 from i to j in G if and only if there is an edge $(i, j) \in o^+$ in G^+.

Notice that, given a DAG $G = \langle E, o\rangle$ and its transitive closure G^+, for every $T \subseteq E$, the subgraph of G^+ induced by T is a transitive closure (we will exploit this property of transitive closure in Section 3).

In the following, we will use the notations $o \uparrow (e_1, e_2)$ and $o \downarrow (e_1, e_2)$ as shorthands for $(o \cup \{(e_1, e_2)\})^+$ and $(o \cup \{(e_1, e_2)\})^-$, respectively.

We 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, $W_E \subseteq O_E$. We will use the letters o and w, possibly subscripted, to denote binary acyclic relations and partial orders, respectively. Clearly, if (E, o) is a directed acyclic graph, then (E, o^+), is a strictly ordered set. 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, $O_E/\sim$ and W_E are isomorphic.

2.2 Basic and Modal Event Calculus

A compact model-theoretic formalization of Kowalski and Sergot's *Event Calculus* (*EC*) [5] has been provided by Cervesato and Montanari in [1]. It distinguishes between time-independent and time-dependent components of *EC*. The time-independent component is captured by means of the notion of *EC-structure*.

Definition 3. (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 incompatibility among properties.

The time-dependent component is formalized by specifying a binary acyclic relation o, called *knowledge state*, on the set of events E.

Given a structure $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ and a knowledge state o, *EC* permits inferring the *maximal validity intervals* (*MVI*s) over which a property p holds uninterruptedly. An *MVI* for p is represented 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 *query language* of *EC* is the set $\mathcal{L}(\mathrm{EC}) = \{p(e_1, e_2) : p \in P \text{ and } e_1, e_2 \in E\}$ of all such property-labeled pairs of events over $\mathcal{H}$. 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 o^+. The elements of $\mathcal{L}(\mathrm{EC})$ are interpreted relative to the set W_E (hereinafter denoted $W_\mathcal{H}$) of partial orders among events in E.

Definition 4. (Intended model of EC)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be a EC-structure and $w \in W_\mathcal{H}$ be the transitive closure of a knowledge state o. 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. $(e_1, e_2) \in w$; **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

$$\exists e \in E\ \exists q \in P\ ((e_1, e) \in w \wedge (e, e_2) \in w \wedge (e \in [q\rangle \vee e \in \langle q]) \wedge (]p, q[\vee p = q))$$

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 [1]. The *Modal Event Calculus* (*MEC*) [1] 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 $\mathcal{L}(\mathrm{MEC})$ 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 5. (Intended model of MEC)

Let $\mathcal{H}$ be an EC-structure and $\upsilon_{\mathcal{H}}$ be the propositional valuation defined as in Definition 4. 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 φ at w with respect to $\mathcal{I}_{\mathcal{H}}$, denoted by $\mathcal{I}_{\mathcal{H}}; w \models \varphi$, is defined as follows:

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

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 w. We denote with $MVI(w)$, $\Box MVI(w)$ and $\Diamond MVI(w)$ the sets of *MVI*s, necessary *MVI*s and possible *MVI*s with respect to w, respectively.

Given $w \in W_{\mathcal{H}}$ and $p \in P$, let $S(w)$ 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 $C(w)$ 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. Formally,

$$\begin{array}{l}
S(w) = \{p(e_1, e_2).\ \forall e \in E\ (\exists q \in P\ (e \in [q\rangle\ \vee\ e \in \langle q])\ \wedge\ (]p, q[\vee p = q)) \rightarrow \\
\qquad (((e, e_1) \in w\ \vee\ (e_1, e) \in w)\ \wedge\ ((e, e_2) \in w\ \vee\ (e_2, e) \in w)) \\
C(w) = \{p(e_1, e_2).\ e_1 \in [p\rangle\ \wedge\ e_2 \in \langle p]\ \wedge\ (e_1, e_2) \notin w\ \wedge\ (e_2, e_1) \notin w))
\end{array}$$

It easily follows that the sets $\Box MVI(w)$ and $\Diamond MVI(w)$ can be alternatively defined in terms of the sets $MVI(w)$, $S(w)$ and $C(w)$ as follows:

Corollary 6. *Let* $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ *be an* EC*-structure and* $w \in W_{\mathcal{H}}$ *be the transitive closure of a knowledge state* o*. It holds that:*

$$\Box MVI(w) = MVI(w) \cap S(w) \text{ and } \Diamond MVI(w) = MVI(w) \cup C(w)$$

3 A generate-only strategy for EC

In this section, we first revise the *generate-only* strategy for *MVI*s computation in the special case of *EC*-structures whose set of properties is a singleton, which has been originally proposed by Chittaro et al. in [2]. Then, we generalize it to the (general) multiple-property case. We provide a high-level description of the algorithms for update and query processing and prove their soundness and completeness with respect to the given *EC* semantics.

3.1 The single-property case (revisited)

In this section, we describe a generate-only strategy for *MVI*s computation in *EC* in the single-property case. Let $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ be an *EC*-structure, with $P = \{p\}$, $o^- \in O_E$ (hereinafter denoted $O_{\mathcal{H}}$) be the transitive reduction of a knowledge state o, and (e_1, e_2) be a pair of events. The addition of the ordered pair (e_1, e_2) to o^- is dealt with as follows (update processing): (e_1, e_2) is first checked for consistency and redundancy with respect to o^-. If (e_1, e_2) is neither inconsistent ($(e_2, e_1) \notin o^+$) nor redundant ($(e_1, e_2) \notin o^+$), then o^- is replaced by $o^- \downarrow (e_1, e_2)$. The set $o^- \downarrow (e_1, e_2)$, which can be proved to be the transitive reduction of $o \cup \{(e_1, e_2)\}$, is obtained as follows: first, the ordered pair (e_1, e_2) is added to o^-; then, the set of nodes from which e_1 is accessible (let us denote this set by $\mathrm{Pred}(e_1)$) and the set of nodes which are accessible from e_2 (let us denote this set by $\mathrm{Succ}(e_2)$) are computed; finally, all pairs (e', e'') of o^- such that $e' \in \mathrm{Prec}(e_1)$ and $e'' \in \mathrm{Succ}(e_2)$ are deleted from $o^- \cup \{(e_1, e_2)\}$.

if $(e_1, e_2) \notin o^+$ **and** $(e_2, e_1) \notin o^+$ **then**
  $o^- \leftarrow o^- \cup \{(e_1, e_2)\}$
  put in $Prec(e_1)$ the nodes from which e_1 is accessible
  put in $Succ(e_2)$ the nodes accessible from e_2
  for each $e' \in Prec(e_1)$ **do**
    for each $e'' \in Succ(e_2)$ **do**
      if $(e', e'') \in o^-$ **then**
        $o^- \leftarrow o^- \setminus \{(e', e'')\}$

Given two events e_i and e_j, testing whether $(e_i, e_j) \in o^+$ or not can be performed by visiting depth-first the subgraph of (E, o^-) generated by e_i and searching for the node e_j. The set $\mathrm{Succ}(e_2)$ can be computed by exploiting a depth-first visit of the subgraph of $(E, o^- \cup \{(e_1, e_2)\})$ generated by e_2 and retrieving all the visited nodes. In order to compute the set $\mathrm{Prec}(e_1)$, we first replace each $(e', e'') \in o^- \cup \{(e_1, e_2)\}$ by (e'', e'); then, we executed a depth-first visit of the subgraph (of the resulting graph) generated by e_1 and retrieve all the visited nodes.

The set of *MVI*s for p can be easily derived from the transitive reduction o^- of the given knowledge state o. Let us define a p-edge of o^- as any edge $(e_1, e_2) \in o^-$ such that e_1 initiates p and e_2 terminates p. Given an *EC*-structure $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$, with $P = \{p\}$, and a knowledge state o, the set of *MVI*s for p consists of all and only the p-edges of o^-; thus, query processing in the single-property case reduces to the retrieval of the p-edges of o^-.

$MVI \leftarrow \emptyset$
for each $(e_1, e_2) \in o^-$ **do**
  if $e_1 \in [p\rangle$ **and** $e_2 \in \langle p]$ **then**
    $MVI \leftarrow MVI \cup \{p(e_1, e_2)\}$
return MVI

The following theorem proves that the proposed generate-only strategy is sound and complete with respect to the given semantics of *EC*.

Theorem 7. *The proposed generate-only strategy for* MVI*s computation in* EC *in the single-property case is sound and complete.*

3.2 The multiple-property case

In this section, we generalize the proposed generate-only strategy for *MVI*s computation in *EC* to the multiple-property case. Let $\mathcal{H}$ be an *EC*-structure, w be the transitive closure of a knowledge state o, and (e_1, e_2) be a pair of events. The addition of (e_1, e_2) to w is dealt with in three steps as follows (update processing):

1. if $(e_2, e_1) \notin w$ and $(e_1, e_2) \notin w$, then w is replaced by $w \uparrow (e_1, e_2)$;
2. for every property $p \in P$, the subgraph w_p induced by the set of events interfering with p, that is, events that initiate or terminate p or a property which is incompatible with p, is extracted from $w \uparrow (e_1, e_2)$;
3. for every property $p \in P$, the transitive reduction w_p^- of the graph w_p is computed.

Whenever both $(e_2, e_1) \notin w$ and $(e_1, e_2) \notin w$, the update procedure computes $w \uparrow (e_1, e_2)$ as follows: first, the edge (e_1, e_2) is added to w; then, for every pair of events $e', e'' \in E$ such that $(e', e_1) \in w$, $(e_2, e'') \in w$, and $(e', e'') \notin w$, the edge (e', e'') is added to $w \cup (e_1, e_2)$. It is worth noting that, since w is transitive, the set of predecessors (resp. successors) of e_1 (resp. e_2) coincides with the set of nodes from which e_1 is accessible (resp. accessible from e_2). Then, for every property $p \in P$, the subgraph w_p is computed by extracting the edges of $w \uparrow (e_1, e_2)$ such that both their endpoints interfere with p. Finally, the transitive reduction w_p^- of w_p is computed by using one of the standard algorithms, e.g. [4, 6].

if $(e_1, e_2) \notin w$ **and** $(e_2, e_1) \notin w$ **then**
  $w \leftarrow w \cup \{(e_1, e_2)\}$
  put in $I\_Pred(e_1)$ the predecessors of e_1
  put in $I\_Succ(e_2)$ the successors of e_2
  for each $e' \in I\_Pred(e_1)$ **do**
    for each $e'' \in I\_Succ(e_2)$ **do**
      if $(e', e'') \notin w$ **then**
        $w \leftarrow w \cup \{(e', e'')\}$

for each $p \in P$ **do**
  $w_p \leftarrow \emptyset$
  for each $(e', e'') \in w$ **do**
    if both e' **and** e'' interfere with p **then**
      $w_p \leftarrow w_p \cup \{(e', e'')\}$

for each $p \in P$ **do**
  compute the transitive reduction w_p^-

The set of *MVI*s for a given property p consists of all and only the p-edges of w_p^-. Hence, for every property p, query processing reduces to the retrieval of the p-edges of w_p^-.

$MVI \leftarrow \emptyset$
for each $p \in P$ **do**
  for each $(e_1, e_2) \in w_p^-$ **do**
    if $e_1 \in [p\rangle$ **and** $e_2 \in \langle p]$ **then**
      $MVI \leftarrow MVI \cup \{p(e_1, e_2)\}$
return MVI

The next theorem shows that the above generate-only strategy for the general multiple-property case is sound and complete with respect to the given semantics of *EC*.

Theorem 8. *The proposed generate-only strategy for* MVI*s computation in* EC *is sound and complete.*

Proof. Let $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ be an *EC*-structure and w be the transitive closure of a knowledge state o. To prove that the proposed strategy is sound, we must show that if (e_1, e_2) is a p-edge of w_p^-, then $p(e_1, e_2)$ is an *MVI* for p with respect to w and $\mathcal{H}$. The proof is by contradiction. If $p(e_1, e_2)$ is not an *MVI*, then one of the following propositions must hold: e_1 does not initiate p, e_2 does not terminate p, $(e_1, e_2) \notin w$, or there exists an interrupting event e for p that occurs between e_1 and e_2. If e_1 does not initiate p or e_2 does not terminate p, then (e_1, e_2) is not a p-edge. If $(e_1, e_2) \notin w$, then $(e_1, e_2) \notin w_p^-$, since $w_p^- \subseteq w$, and thus (e_1, e_2) is not a p-edge of w_p^-. Finally, if there exists an interrupting event e for p such that both $(e_1, e) \in w$ and $(e, e_2) \in w$, then there exist a path $e_1 \rightsquigarrow e$ and a path $e \rightsquigarrow e_2$ in w_p^-. Hence, the edge (e_1, e_2) is a transitive one, and thus it does not belong to w_p^-. This allows us to conclude that (e_1, e_2) is not a p-edge of w_p^-.

To prove that the proposed strategy is complete, we must show that if $p(e_1, e_2)$ is an *MVI* for p with respect to w and $\mathcal{H}$, then (e_1, e_2) is a p-edge of w_p^-. By hypothesis, e_1 initiates p, e_2 terminates p, and $(e_1, e_2) \in w$. It follows that (e_1, e_2) is a p-edge of w_p. Moreover, since there are no interrupting events for p that occur between e_1 and e_2, the edge (e_1, e_2) is the unique path from e_1 to e_2 in w_p. This implies that the edge (e_1, e_2) is not transitive, and thus it is a p-edge of w_p^-. □

It is easy to devise two efficient algorithms, that respectively compute the necessary and possible *MVI*s, exploiting Corollary 6 and taking advantage of the algorithm for basic *MVI*s proposed in Section 3.

4 Complexity analysis

In this section, we analyze the worst-case computational complexity of the proposed algorithms for update and query processing in *EC* and *MEC*.

Given an *EC*-structure $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ and a knowledge state $o \in O_{\mathcal{H}}$, we determine the complexity of computing the set of *MVI*s with respect to o and $\mathcal{H}$, that is, of determining the set of formulas $p(e_1, e_2)$ such that $w \models p(e_1, e_2)$,

by means of the proposed generate-only strategy. We assume that the set of events E can grow arbitrarily, while the set P of relevant properties characterizes the considered application domain and thus it is fixed once and for all. As a consequence, we choose the number n of events in E as the size of $\mathcal{H}$, and consider the number of properties as a constant. We measure the complexity in terms of the size n of $\mathcal{H}$ and the size m of the knowledge state o, or the size m^- (resp. m^+) of its transitive reduction o^- (resp. closure o^+)[2]. We assume P to be lexicographically sorted. Furthermore, we assume that the knowledge state o as well as the sets $[p\rangle$ and $\langle p]$, for every property $p \in \mathcal{H}$, are (maintained) sorted. Under such an assumption, given an event e and a property p, the tests $e \in [p\rangle$ and $e \in \langle p]$ cost $\mathcal{O}(\log n)$, while given two distinct events e_1 and e_2, the test $(e_1, e_2) \in o$ costs $\mathcal{O}(\log m)$. Finally, the test $(e_1, e_2) \in o^+$ can be performed in $\mathcal{O}(m+n)$ by executing a depth-first visit of the subgraph of (E, o) generated by e_1.

The complexity of the generate-only strategy for the single-property case in *EC* is given by the following theorem.

Theorem 9. *Let $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ be an* EC*-structure, with $P = \{p\}$, o^- be the transitive reduction of a knowledge state o, and (e_1, e_2) be a pair of events. The complexity of update processing is $\mathcal{O}(n^2 \cdot \log m^-)$, while the complexity of query processing is $\mathcal{O}(m^- \cdot \log n)$, where m^- is the cardinality of o^-.*

The following theorem determines the complexity of the generate-only strategy for the multiple-property case in *EC*.

Theorem 10. *Let $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ be an* EC*-structure, w be the transitive closure of a knowledge state o and (e_1, e_2) be a pair of events. The complexity of update processing is $\mathcal{O}(n \cdot m^- + n^2 \cdot \log m^+)$, while the complexity of query processing is $\mathcal{O}(m^- \cdot \log n)$, where m^+ is the cardinality of $\hat{w} = w \uparrow (e_1, e_2)$ and m^- is the cardinality of $\hat{w}^-$.*

Proof. Update processing is performed in three steps. At the first step, it controls that neither (e_1, e_2) nor (e_2, e_1) belong to w. If this is the case, it computes the set $\hat{w} = w \uparrow (e_1, e_2)$. The tests $(e_1, e_2) \notin w$ and $(e_2, e_1) \notin w$ cost $\mathcal{O}(\log m^+)$. The set $\hat{w}$ is computed as follows: first, the edge (e_1, e_2) is added to w; then, for every pair of events $e', e'' \in E$ such that $(e', e_1) \in w$, $(e_2, e'') \in w$, and $(e', e'') \notin w$, the edge (e', e'') is added to $w \cup (e_1, e_2)$. The sets of predecessors and successors of a given node have both cardinality $\mathcal{O}(n)$, and the addition of (e', e'') to $w \cup (e_1, e_2)$ costs $\mathcal{O}(\log m^+)$; hence, the complexity of computing $\hat{w}$ is $\mathcal{O}(n^2 \cdot \log m^+)$. The second step consists in the extraction of $\hat{w}_p$ from $\hat{w}$, for every property p. For every property p, $\hat{w}_p$ contains the edges (e', e'') of $\hat{w}$ such that both e' and e'' interfere with p. Since each "interference" test costs $\mathcal{O}(\log n)$, this step has complexity $\mathcal{O}(m^+ \cdot \log n)$. The last step is the computation of the transitive reduction $\hat{w}_p^-$, for every property p. Since $\hat{w}_p$ is acyclic, $\hat{w}_p^-$ can be computed in $\mathcal{O}(n \cdot m^-)$. The resulting cost of update processing is thus $\mathcal{O}(n \cdot m^- + m^+ \cdot \log n + n^2 \cdot \log m^+) =$

[2] It is well-known that m, m^-, and m^+ are $\mathcal{O}(n^2)$.

$\mathcal{O}(n \cdot m^- + n^2 \cdot \log m^+)$. Query processing consists in picking up, for every property p, the p-edges of $\hat{w}_p^-$. Since the cardinality of $\hat{w}_p^-$ is $\mathcal{O}(m^-)$ and verifying whether or not an edge is a p-edge costs $\mathcal{O}(\log n)$, the complexity of query processing is $\mathcal{O}(m^- \cdot \log n)$. □

5 Conclusions

In this paper, we have shown how the graph-theoretic notions of transitive closure and reduction of a directed acyclic graph can be successfully exploited to efficiently reason about partially ordered events in *EC* and *MEC*. Even though we developed our solution in the (Modal) Event Calculus context, we expect it to be applicable to any formalism for reasoning about partially ordered events.

The complexities of update and query processing of the standard generate-and-test strategy for *EC* are $\mathcal{O}(1)$ and $\mathcal{O}(n^5)$, respectively. The alternative generate-only strategy we outlined moves computational complexity from query to update processing and features an absolute improvement of performance. In particular, whenever the size of the transitive reduction is $\mathcal{O}(n)$ (we expect this case to be the most frequent) update and query processing can be performed in $\mathcal{O}(n^2 \cdot \log n)$ and $\mathcal{O}(n \cdot \log n)$, respectively (the factor $\log n$ can actually be eliminated by using suitable hashing techniques).

References

1. 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.
2. L. Chittaro, A. Montanari, and I. Cervesato. Speeding up temporal reasoning by exploiting the notion of kernel of an ordering relation. In *Proc. of the 2nd International Workshop on Temporal Representation and Reasoning — TIME'95*, pages 73–80, Melbourne Beach, FL, 26 April 1995.
3. Thomas Dean and Mark Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36:375–399, 1988.
4. A. Goralcikova and V. Koubek. A reduct and closure algorithm for graphs. In *Proc. of the 8th Symposium on Mathematical Foundations of Computer Science. LNCS 74*, pages 301–307, Olomouc, CZ, 1979. Springer.
5. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.
6. K. Simon. An improved algorithm for transitive closure on acyclic digraphs. *Theoretical Computer Science*, 58(1-3):325–346, 1988.
7. J. van Leeuwen. Graph algorithms. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity*, pages 525–632. Elsevier, 1990.