On the Soundness of Attack Trees

Maxime Audinot and Sophie Pinchinat

IRISA, Campus de Beaulieu, 35042 Rennes Cedex, FRANCE.
E-mails:{maxime.audinot,sophie.pinchinat}@irisa.fr

Abstract. We formally define three notions of soundness of an attack tree w.r.t. the system it refers to: *admissibility*, *consistency*, and *completeness*. The system is modeled as a labeled transition system and the attack is provided with semantics in terms of paths of the transition system. We show complexity results on the three notions of soundness, and the influence of the operators that are in the attack tree (see the recap in Figure 5).

1 Introduction

Attack trees [8, 5, 4] are graphical representations of sets of attacks described in a hierarchical manner. The hierarchy is reflected in the structure of the tree, whose internal nodes represent abstract attack goals, and leaf nodes represent atomic goals. Internal nodes of an attack tree have extra information, namely the combinator (or operator) which expresses how the goal of the current node decomposes into the combination of its children goals. Classic operators are the "or" operator, the "sequential" operator, and the "and" operator.

Attack trees are a common tool used in risk analysis. The tree is used to describe the attacks to which of a system is vulnerable. First, an attack tree is constructed from a model of the system, and then it is analyzed for quantitative results, like computing the likelihood of an attack. In this paper, we focus on the qualitative part of attack trees, because our trees can be post-processed to take likelihood into account by adding weigths to the leafs and propagating them.

There are different ways of defining the semantics of attack trees, which unsurprisingly strongly relies on the semantics of the set of operators. In [5], the focus is put on quantitative interpretations: atomic goals are given values in a domain, then, via the operators' semantics, a bottom-up computation yields a value at the (root node of the) tree that corresponds to, *e.g.* the length of the shortest attack, the highest probability to achieve an attack, etc.

In this contribution, we propose various semantics of attack trees that enable us to interpret them in the context of the system they refer to. This is strongly motivated by the nature of the top-down manual design of attack trees by practitioners, where the leaves a tree are iteratively refined into a combination of sub-nodes. To our knowledge, this issue has not been addressed in the literature.

In our setting, the system the tree refers to is a standard transition system $\mathcal{S}$ labeled over a set of atomic propositions Prop. It represents the operational semantics of some domain, as done in [7] for military buildings, or in [6] for socio-technical systems, leaving aside quantitative aspects (likelihood, time, cost). We describe the *attack goal* of a node by an expression $\iota \blacktriangleright \gamma$, where $\iota, \gamma \in \mathrm{Prop}$ are atomic propositions that denote respectively the preconditions and post-conditions of the goal (in the spirit of automated planning approaches). A natural system-based denotational path semantics is given to an attack goal $\iota \blacktriangleright \gamma$, where ι and γ are atomic propositions: the denoted set of paths is composed of all paths of the fixed transition system $\mathcal{S}$ that start from a state labeled by the precondition ι and that end in a state labeled by the post-condition γ. The internal nodes of an attack tree carry an attack goal, together with the operator that describes its decomposition into sub-goals[1], hence a pair $(\iota \blacktriangleright \gamma, \Box)$; we call such an internal node a $\Box$-node. In this paper, we let $\Box$ range over $\{\ovee, \ogreaterthan, \owedge\}$ for the “or”, the “sequential and”, and the “and” operators respectively. In our graphical representations of attack trees (see Figure 2 on Page 8), the shapes of the nodes emphasize the operator associated to the node: $\ovee$-nodes are represented with an ellipse, $\ogreaterthan$-nodes are represented with pentagons pointing rightwards, and $\owedge$-nodes are represented with rectangles, and the leaf nodes are represented with rounded corners rectangles.

In this paper, we address the soundness of an attack tree in terms of the relationship between an internal node $(\iota \blacktriangleright \gamma, \Box)$ and the list of its children nodes $(\iota_1 \blacktriangleright \gamma_1, \Box_1)$,..., $(\iota_n \blacktriangleright \gamma_n, \Box_n)$ (from left to right). To do

[1] the children of the internal node.

so, we compare[2] the set of paths denoted by $\iota \blacktriangleright \gamma$ with the $\Box$-combination of the sets of paths denoted by the children $\iota_i \blacktriangleright \gamma_i$ of that node.

We introduce three notions of soundness for attack trees w.r.t. the transition system: *admissibility*, *consistency*, and *completeness*. Admissibility captures the approach where practitioners decompose the main goal into a structured goal some of whom achievements are also an achievements of the main goal. Consistency expresses that the proposed decomposition of the main goal guarantees its achievement. Finally, the intent of completeness is a complete characterization of the main goal in terms of the proposed decomposition.

The three notions of soundness are defined by comparing the two sets of paths denoted by $\iota \blacktriangleright \gamma$ and the $\Box$-combination of the sets of paths denoted by the children. We use the three natural comparisons between sets, namely equality, inclusion, and non-empty intersection. Each notion of soundness entails a decision problem, of whether a given attack tree is sound or not w.r.t. the transition system it refers to. We establish complexity results on the three notions of soundness, and with regards to the kinds of operators that are allowed. We show that the admissibility problem is in P for the operators $\ovee$ and $\otriangleright$, but becomes NP-complete for the operator $\owedge$. Next, we prove that the consistency problem is in P for the operators $\ovee$, co-NP[3] for the operator $\otriangleright$ and co-NP-complete for the operator $\owedge$. The completeness problem is in co-NP for the operators $\ovee$ and $\otriangleright$, and in Π_2^P for the operator $\owedge$. Recall that Π_2^P is a complexity class of the polynomial hierarchy [10] composed of languages whose complement is in Σ_2^P, or equivalently $\mathrm{NP}^{\mathrm{NP}}$, that are languages captured by a non-determinitsic polynomial-time algorithm which can call a non-determinitsic polynomial-time subroutine[4].

The paper is organized as follows: In Section 2, we present preliminaries notions used in the rest of the paper. In Section 3, we present transitions systems and formal attack goals, and their paths properties. In Section 4, we present attack trees, and the three soundness completeness, consistency and admissibility. In Section 5, we show the complexity re-

[2] see further for details.
[3] that is the *negative instances* of the decision problem, *i.e.* those for which the answer is "no", are fully characterized by a polynomial-time non-deterministic algorithm.
[4] which is classically called an *oracle*.

sults for the three soundness. In Section 6, we discuss the complexity result and conjecture about the harness that are not established yet.

2 Preliminaries

For $i, j \in \mathbb{N}$, we denote by $[i; j]$ the *interval* of integers ranging over $\{i, i+1, \ldots j\}$. For a finite set X, 2^X is the powerset of X, $|X|$ is the cardinal of X, X^* is the set of finite sequences of elements of X. For a binary relation R over a set X ($R \subseteq X \times X$), we say that R is *left-total* if for every $x \in X$, there exists $y \in X$ such that $(x, y) \in R$. We denote by R^* the reflexive and transitive closure of R.

We recall that P is the class of decision problems[5] that can be solved by a deterministic polynomial-time algorithm, that NP is the class of decision problems that can be solved by a non-deterministic polynomial-time algorithm, and co-NP is the class of decision problems whose complementary problem[6] is in NP. As a typical representative of the class NP, we will consider the classical decision problem `SAT` (We refer to [3] for these classic classes of complexity). We end with the class Π_2^P of the polynomial hierarchy which captures the decision problems whose negative instances can be characterized by a non-determinitsic polynomial-time algorithm which can call a non-determinitsic polynomial-time subroutine[7]. We refer to [10] for details on the polynomial hierarchy.

3 Transition systems and attack goals

In this section, we define transition systems, attack goals and the semantics of the operators $\{\ovee, \ogreaterthan, \owedge\}$.

[5] the answer is "Yes/No".
[6] the answers "Yes/No" are swapped.
[7] which is classically called an *oracle*.

3.1 Transition systems

Without loss of generality and for technical reasons, transition systems will carry no actions, but instead have all the necessary information in their states via a labeling by atomic propositions.

Definition 1 (Transition system). *Let* Prop *be a finite set of atomic propositions. A* transition system *over* Prop *is a tuple* $\mathcal{S} = (S, \rightarrow, \lambda)$*, where:*

- S *is the finite set of* states*,*
- $\rightarrow \subseteq S \times S$ *is the* transition relation *of the system (which is assumed left-total*[8]*),*
- $\lambda : \text{Prop} \rightarrow 2^S$ *is the* valuation *function.*

The size of $\mathcal{S}$ *is* $|\mathcal{S}| = |S| + |\rightarrow|$.

Let $S' \subseteq S$ *be a sub-set of states. We let* $Post^*_{\mathcal{S}}(S')$ *be the set of states that are reachable from some state of* S'*, and* $Pre^*_{\mathcal{S}}(S')$ *be the set of states that are co-reachable from some state of* S'*. Formally,*

- $Post^*_{\mathcal{S}}(S') := \{s \in S \mid \textit{there is some } s' \in S' \textit{ such that } s' \rightarrow^* s\}$
- $Pre^*_{\mathcal{S}}(S') := \{s \in S \mid \textit{there is some } s' \in S' \textit{ such that } s \rightarrow^* s'\}$.

We will use the following running example:

Example 1.
The set Prop_e is $\{i, f, m_1, m_2, e_1, e_2, pre_a, post_a, pre_b, post_b, pre_c, post_c\}$, the system $\mathcal{S}_e = (S_e, \rightarrow_e, \lambda_e)$ over Prop_e, whose graphical representation is given in Figure 1, is formally defined by: $S_e = \{s_i\}_{0 \le i \le 6}$, where the transition relation $\rightarrow_e$ contains the pairs (s_0, s_1), (s_0, s_2), (s_1, s_3), (s_2, s_3), (s_2, s_4), (s_3, s_5), (s_4, s_6). Finally, we let $\lambda_e(i) = \lambda_e(pre_a) = \{s_0\}$, $\lambda_e(f) = \{s_5, s_6\}$, $\lambda_e(m_1) = \lambda_e(post_a) = \{s_1, s_2\}$, $\lambda_e(m_2) = \{s_3, s_4\}$, $\lambda_e(e_1) = \{s_5\}$, $\lambda_e(e_2) = \{s_6\}$, $\lambda_e(pre_b) = \{s_1, s_2, s_4\}$, $\lambda_e(post_b) = \{s_3, s_6\}$, $\lambda_e(pre_c) = \{s_2, s_3\}$, and finally, $\lambda_e(post_c) = \{s_4, s_5\}$. Also, $Pre^*_{\mathcal{S}}(\{s_3\}) = \{s_0, s_1, s_2, s_3\}$ and $Post^*_{\mathcal{S}_e}(\{s_1, s_6\}) = \{s_1, s_3, s_5, s_6\}$.

[8] this is classic and it is no loss of generality.

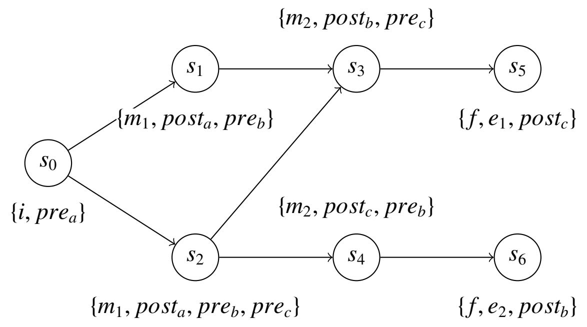


Fig. 1. Example of transition system: $\mathcal{S}_e$.

Definition 2 (Paths, elementary paths, factors). *A* path *in a system $\mathcal{S}$ is a sequence of states of the form $\pi = s_0 s_1 \dots s_n \in S^*$ for some $n \in \mathbb{N}$, such that for all $k \in [0; n-1], (s_k, s_{k+1}) \in \rightarrow$. An* elementary path *is a path $s_0 s_1 \dots s_n$ where $\forall k \neq k' \in [0; n], s_k \neq s_{k'}$ (*i.e. *there is no cycles). We denote by $\Pi(\mathcal{S})$ the set of paths of $\mathcal{S}$. Let $\pi = s_0 \dots s_n \in \Pi(\mathcal{S})$. The* length *of π is n, written $|\pi|$. A* factor *of π is a sequence $s_i \dots s_j$ for some $0 \leq i \leq j \leq n$, that will be denoted by $\pi[i; j]$. The interval $[i; j]$ is an* anchoring interval*, or simply an* anchoring*, of the factor $\pi[i; j]$ in π.*

We define two notions of decomposition of a path that reflect the refinement of attack tree nodes. Both are based on factors.

Definition 3 (Sequential and parallel decomposition of paths). *Let $\pi \in \Pi(\mathcal{S})$ be a path. A sequence $\pi_1. \dots .\pi_k \in \Pi(\mathcal{S})^*$ of paths is a* sequential decomposition *of π if each π_j is a factor of π, there are ordered anchorings of the π_j's that form a tiling of the interval $[0; |\pi|]$. In particular, this anchoring of π_1 is of the form $[0; y]$ and this anchoring of π_k is of the form $[x; |\pi|]$. A set of paths $\{\pi_1, \dots, \pi_k\}$ is a* parallel decomposition *of $\pi \in \Pi(\mathcal{S})$ if each π_j is a factor of π, and the anchorings of the paths π_j cover the interval $[0; |\pi|]$. Notice that a sequential decomposition is a particular case of a parallel decomposition.*

Example 2. Consider the path $\pi = s_0 s_2 s_3 s_5$ in system $\mathcal{S}_e$. The sequence $(s_0 s_2).(s_2 s_3 s_5)$ is a sequential decomposition of π, where the anchoring of $s_0 s_2$ is (unique and equal to) $[0; 1]$ and the unique anchoring of $s_2 s_3 s_5$ is $[1; 3]$. The set $\{s_2 s_3 s_5, s_0 s_2 s_3\}$ is a parallel decomposition of π, where the anchoring of $s_2 s_3 s_5$ is $[1; 3]$ and the anchoring of $s_0 s_2 s_3$ is $[0; 2]$.

For the section 3.2 and 4, we fix a transition system $\mathcal{S} = (S, \rightarrow, \lambda)$.

3.2 Attack goals

Attack goals are descibed in a formal language meant to specify attack objectives that internal nodes of an attack tree naturally carry.

Definition 4 (Attack goals). *An* attack goal *is an expression of the form either $\iota \blacktriangleright \gamma$, or a term of the form $(\iota_1 \blacktriangleright \gamma_1) \Box (\iota_2 \blacktriangleright \gamma_2) \Box \dots (\iota_n \blacktriangleright \gamma_n)$, where $\Box \in \{\ovee, \ogreaterthan, \owedge\}$ and $\iota, \iota_1, \dots \iota_n, \gamma, \gamma_1, \dots \gamma_n \in$* Prop.

Example 3. $i \blacktriangleright f$, $(i \blacktriangleright e_1) \ovee (i \blacktriangleright e_2)$ and $(i \blacktriangleright post_a) \ogreaterthan (post_a \blacktriangleright post_c) \ogreaterthan (post_c \blacktriangleright post_b)$ are attack goals, whose interpretation will be given in system $\mathcal{S}_e$ (see Example 4).

Definition 5 (Path semantics of attack goals). *The path semantics of an attack goal t, written $[t]^{path}_{\mathcal{S}}$, is a subset of $\Pi(\mathcal{S})$ defined by: if $t = \iota \blacktriangleright \gamma$, then $[\iota \blacktriangleright \gamma]^{path}_{\mathcal{S}} = \{\pi \in \Pi(\mathcal{S}) \mid \pi(0) \in \lambda(\iota)$ and $\pi(|\pi|) \in \lambda(\gamma)\}$, otherwise we distinguish between the different operators $\Box \in \{\ovee, \ogreaterthan, \owedge\}$ according to:*

$$[(\iota_1 \blacktriangleright \gamma_1) \ovee_n (\iota_2 \blacktriangleright \gamma_2) \ovee_n \dots (\iota_n \blacktriangleright \gamma_n)]^{path}_{\mathcal{S}} = \bigcup_{1 \le i \le n} [\iota_i \blacktriangleright \gamma_i]^{path}_{\mathcal{S}}$$

$$[(\iota_1 \blacktriangleright \gamma_1) \ogreaterthan_n (\iota_2 \blacktriangleright \gamma_2) \ogreaterthan_n \dots (\iota_n \blacktriangleright \gamma_n)]^{path}_{\mathcal{S}} = \{\pi \mid \textit{there is a decomposition } \pi_1.\pi_2 \dots .\pi_n \textit{ of } \pi \textit{ and each } \pi_i \in [\iota_i \blacktriangleright \gamma_i]^{path}_{\mathcal{S}}\}$$

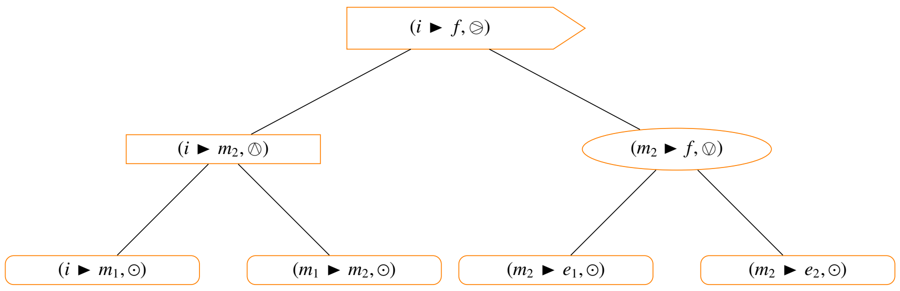
$$[(\iota_1 \blacktriangleright \gamma_1) \owedge_n (\iota_2 \blacktriangleright \gamma_2) \owedge_n \dots (\iota_n \blacktriangleright \gamma_n)]^{path}_{\mathcal{S}} = \{\pi \mid \textit{there is a parallel decomposition } \{\pi_1, \pi_2, \dots, \pi_n\} \textit{ of } \pi \textit{ and each } \pi_i \in [\iota_i \blacktriangleright \gamma_i]^{path}_{\mathcal{S}}\}$$

Example 4. The attack goals of Example 3 have the following path semantics:

$$[i \blacktriangleright f]^{\text{path}}_{\mathcal{S}_e} = \{\pi \in \Pi(\mathcal{S}_e) \mid \pi(0) = s_0 \text{ and } \pi(|\pi| \in \{s_5, s_6\}\}$$
$$[(i \blacktriangleright e_1) \ovee_2 (i \blacktriangleright e_2)]^{\text{path}}_{\mathcal{S}_e} = [i \blacktriangleright f]^{\text{path}}_{\mathcal{S}_e}$$
$$[(i \blacktriangleright post_a) \ogreaterthan_3 (post_a \blacktriangleright post_c) \ogreaterthan_3 (post_c \blacktriangleright post_b)]^{\text{path}}_{\mathcal{S}_e} = \{s_0 s_2 s_4 s_6\}$$

4 Attack trees

We define the set of attack trees over a set Prop of atomic propositions. In addition to the classical branching structure with nodes typed by a operator, we decorate each node with an attack goal $\iota \blacktriangleright \gamma$, representing the goal of the node. This goal is a formalization of the what is usually written in plain text in nodes of classical attack trees.



$(i \blacktriangleright f, \ogreaterthan)$
$(i \blacktriangleright m_2, \owedge)$
$(m_2 \blacktriangleright f, \ovee)$
$(i \blacktriangleright m_1, \odot)$
$(m_1 \blacktriangleright m_2, \odot)$
$(m_2 \blacktriangleright e_1, \odot)$
$(m_2 \blacktriangleright e_2, \odot)$

Fig. 2. The attack tree T_e.

An *attack tree* (over Prop) is either a leaf of the form $(\iota \blacktriangleright \gamma, \odot)$ or a composed tree of the form $(\iota \blacktriangleright \gamma, \Box)(T_1, T_2 \ldots T_n)$, where $\iota, \gamma \in \mathrm{Prop}$, $\Box \in \{\ovee, \ogreaterthan, \owedge\}$, $n \geq 2$, and T_1, T_2, l. . . , T_n are attack trees.We call the main node of a non-leaf tree a $\Box$-*node* whenever it is of the form $(\iota \blacktriangleright \gamma, \Box)(T_1, T_2 \ldots T_n)$.

The *path semantics* of an attack tree $(\iota \blacktriangleright \gamma, \Box)(T_1, T_2 \ldots T_n)$ is naturally defined as $[\iota \blacktriangleright \gamma]^{\mathrm{path}}_{\mathcal{S}} \subseteq \Pi(\mathcal{S})$.

Example 5. Figure 2 represents the attack tree T_e over Prop_e where:

$$\begin{aligned} T_e = (i \blacktriangleright f, \ogreaterthan)((i \blacktriangleright m_2, \owedge)((i \blacktriangleright m_1, \odot), (m_1 \blacktriangleright m_2, \odot)),\\ (m_2 \blacktriangleright f, \ovee)((m_2 \blacktriangleright e_1, \odot), (m_2 \blacktriangleright e_2, \odot))) \end{aligned}$$

Another example of such an attack tree, using an $\owedge$-node, is $(i \blacktriangleright f, \owedge)(pre_a \blacktriangleright post_a, \odot), (pre_b \blacktriangleright post_b, \odot), (pre_c \blacktriangleright post_c, \odot))$.

For example, the path semantics of T_e is $\Pi(\mathcal{S}_e)$.

We now turn to more subtle semantics for attack trees that enable one to relate a tree with its subtrees, or equivalently an internal node with its children, in terms of their path semantics, hence the explicit reference to the system the attack tree refers to. Our proposal yields three notions of soundness with different interpretations from the point of view of the practitioner. The *admissibility* property means that there is an attack that achieves the the parent node goal that decomposes with the ones of its children nodes (Equation (1)). The *consistency* property means that the combined children node goals yield attacks (if any) that achieve the parent node goal (Equation (2)). Finally, the *completeness* property means that the combined children node goals fully characterize the parent node goal (Equation (3)).

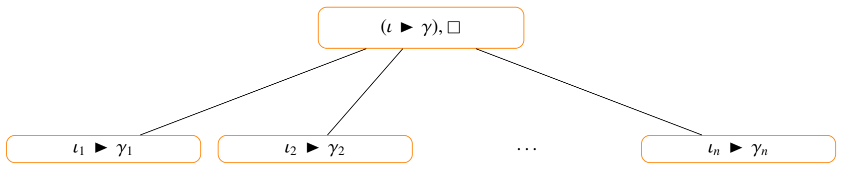

$(\iota \blacktriangleright \gamma), \square$
$\iota_1 \blacktriangleright \gamma_1$
$\iota_2 \blacktriangleright \gamma_2$
$\cdots$
$\iota_n \blacktriangleright \gamma_n$

Fig. 3. A picture for Equation (1).

Definition 6 (Admissibility).
The attack tree $(\iota \blacktriangleright \gamma, \square)(T_1, T_2 \ldots T_n)$ *is* admissible *w.r.t.* $\mathcal{S}$ *either when* $\square$ *is* $\odot$*, or when Equation* (1) *holds, where* $\iota_i \blacktriangleright \gamma_i$ *is the local attack goal of the tree* T_i $(1 \leq i \leq n)$*, see Figure 3.*

$$[\, \square_{i=1}^{n}\, (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{path} \cap [\iota \blacktriangleright \gamma]_{\mathcal{S}}^{path} \neq \emptyset \tag{1}$$

Then, the *consistency* and *completeness* properties are variants of the admissibility property by replacing Equation (1) of Definition 6 by Equation (3) and Equation (2), respectively:

$$[\, \square_{i=1}^{n}\, (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}} \subseteq [\iota \blacktriangleright \gamma]_{\mathcal{S}}^{\text{path}} \tag{2}$$

$$[\, \square_{i=1}^{n}\, (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}} = [\iota \blacktriangleright \gamma]_{\mathcal{S}}^{\text{path}} \tag{3}$$

Remark 1. As Equation (3) entails Equation (2), completeness implies consistency.

For example, the attack tree $(i \blacktriangleright f, \ogreaterthan)((i \blacktriangleright m_2, \odot), (m_2 \blacktriangleright f, \odot))$ is admissible w.r.t. $\mathcal{S}_e$, whereas $(pre_a \blacktriangleright post_a, \ovee)((pre_b \blacktriangleright post_b, \odot), (pre_c \blacktriangleright post_c, \odot))$ is not admissible w.r.t. $\mathcal{S}_e$.

5 The decision problems `ADM`($\mathcal{O}$), `CONS`($\mathcal{O}$), `COMP`($\mathcal{O}$)

We formalize the decision problems `ADM`($\mathcal{O}$), `CONS`($\mathcal{O}$), `COMP`($\mathcal{O}$) respectively related to the notions of admissibility, consistency and completeness, as introduced in Section 4. Let $\mathcal{O} \subseteq \{\ovee, \ogreaterthan, \owedge\}$.

Definition 7. *The* Admissibility problem `ADM`($\mathcal{O}$) *is defined by:*
Input: $\theta = \iota\gamma\iota_1\gamma_1 \ldots \iota_n\gamma_n$ *a sequence of atomic propositions,* $\Box \in \mathcal{O}$ *and* $\mathcal{S}$ *a labeled transition system over* $\{\iota, \gamma, \iota_1, \ldots, \gamma_n\}$.
Output: *"yes" if* $(\iota \blacktriangleright \gamma, \Box)((\iota_1 \blacktriangleright \gamma_1, \odot), (\iota_2 \blacktriangleright \gamma_2, \odot) \ldots (\iota_n \blacktriangleright \gamma_n, \odot))$ *is admissible w.r.t* $\mathcal{S}$*, "no" otherwise.*

We similarly define the decisions problems `CONS`($\mathcal{O}$) and `COMP`($\mathcal{O}$) in a natural way, respectively called the *the consistency problem* and *the completeness problem*. In the sequel, we will denote by $(\theta, \Box, \mathcal{S})$ an instance of `ADM`($\mathcal{O}$), `CONS`($\mathcal{O}$), or `COMP`($\mathcal{O}$), where unless explicitely stated, θ expands into $\iota\gamma\iota_1\gamma_1 \ldots \iota_n\gamma_n$.

5.1 Preliminary complexity results

We first establish useful technical propositions that will be used to prove our main results on complexity for the three decision problems `ADM`($\mathcal{O}$), `CONS`($\mathcal{O}$), and `COMP`($\mathcal{O}$).

Proposition 1. *Given a path* π*, deciding whether or not* $\pi \in [(\iota \blacktriangleright \gamma)]^{path}_{\mathcal{S}}$ *can be done in constant time. As a consequence, deciding whether or not* $\pi \in [\ \ovee_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{path}_{\mathcal{S}}$ *is also in* P.

Proof. For $\pi \in [(\iota \blacktriangleright \gamma)]^{\text{path}}_{\mathcal{S}}$, the only thing to check is $\pi(0) \in \lambda(\iota)$ and $\pi(|\pi|) \in \lambda(\gamma)$. For $\pi \in [\ \ovee_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}}$, it amount to finding $i \in [1; n]$ such that $\pi \in [(\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}}$.

The next two propositions address operators $\ogreaterthan$ and $\owedge$.

Proposition 2. *Given a path π, deciding whether or not $\pi \in [\ \ogreaterthan_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{S}^{path}$ is in* P.

Proof. It sufficient to check that $\pi(0) \in \lambda(\iota_1)$ and $\pi(|\pi|) \in \lambda(\gamma_n)$ and to make a traversal of π that seeks for a sequence of positions $0 \leq y_1 \leq \cdots \leq y_{n-1} \leq |\pi|$ such that $\pi(y_i) \in \lambda(\gamma_i) \cap \lambda(\iota_{i+1})$, for all $i \in [1; n-1]$.

Proposition 3. *Given a path π, deciding whether or not $\pi \in [\ \owedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{S}^{path}$ is in* NP.

Proof. To verify that $\pi \in [\owedge_{i=1}^{n}(\iota_i \blacktriangleright \gamma_i)]_{S}^{\text{path}}$ the algorithm guesses n factors of π, or equivalently their sequence of anchorings $[x_1; y_1], \ldots, [x_n; y_n]$ and checks they provide a parallel decomposition of π. Namely, the algorithm needs to check the following properties: (i) $x_i \leq y_i$, for each $i \in [1; n]$, (ii) for each $x \in [0; |\pi|]$, there exists $i \in [1; n]$ such that $x_i \leq x \leq y_i$, and (iii) $\pi[x_i; y_i] \in [\iota_i \blacktriangleright \gamma_i]_{S}^{\text{path}}$, that is $\pi(x_i) \in \lambda(\iota_i)$ and $\pi(y_i) \in \lambda(\gamma_i)$. By the above propositions, it is clear that Properties (i)-(iii) can be verified in polynomial time.

The two following propositions are helpful in order to bound the size of the paths we will need to guess in our non-deterministic algorithms of Section 5.2.

Proposition 4. *Let $\square \in \{\owedge, \ogreaterthan, \ovee\}$. If $[\ \square_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{S}^{path} \neq \emptyset$, then it contains a path of size smaller than $|S|(2n-1)$. In particular, if $n = 1$, we can consider a path of length at most $|S|$, that is an elementary path.*

Proof. We first consider the case where $\square = \owedge$. Let $\pi \in [\owedge_{i=1}^{n}(\iota_i \blacktriangleright \gamma_i)]_{S}^{\text{path}}$, and let $[x_1; y_1], \ldots, [x_n; y_n]$ be the anchoring intervals of a parallel decomposition of π, such that, for each $1 \leq i \leq n$, $\pi(x_i) \in \lambda(\iota_i)$ and $\pi(y_i) \in \lambda(\gamma_i)$. Let $z_1 \leq \cdots \leq z_{2n}$ be the resulting of sorting the elements $x_1, y_1 \ldots, x_n, y_n$. Notice that the sequence $\pi[z_1; z_2], \pi[z_2; z_3], \ldots, \pi[z_{2n-1}; z_{2n}]$ is a sequential decomposition of π. For $1 \leq i \leq 2n-1$, let π'_i be the elementary path obtained from $\pi[z_i; z_{i+1}]$ by removing the cycles. We have $\left|\pi'_i\right| \leq |S|$. The path π' obtained by the sequential composition of the paths π'_i is in

$[\ \owedge_{i=1}^{n}\ (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\mathrm{path}}$ since the states $\pi(z_i)$ for $i \in [1; 2n]$ are still in π' and in the same order. Then we have $|\pi'| \leq |S|(2n-1)$, which concludes. Regarding the case where $\Box = \ogreaterthan$, it is enough to remark that sequential decomposition is a particular parallel decomposition, and the case $\Box = \ovee$ is obvious as even elementary paths suffice.

Finally, the following more involved Proposition 5 plays a key role in our proofs of Section 5.2. The rest of this section is dedicated to its proof.

Proposition 5. *Given a $\mathcal{S} = (S, \rightarrow, \lambda)$ be a labeled transition system over a set of propositions* Prop $\supseteq \{\iota, \gamma, \iota_1, \ldots, \gamma_n\}$*, it is* NP*-complete to decide whether or not $[\ \owedge_{i=1}^{n}\ (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{path} \neq \emptyset$.*

NP**-easyness:** We describe the non-deterministic algorithm that decides $[\ \owedge_{i=1}^{n}\ (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\mathrm{path}} \neq \emptyset$. This algorithm guesses a path π such that $|\pi| \leq |S|(2n-1)$ (which is sufficient by Proposition 4), and n anchoring intervals $[x_1; y_1], \ldots, [x_n; y_n]$ in π. It then verifies that $\pi \in [\ \owedge_{i=1}^{n}\ (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\mathrm{path}}$, which can be done in polynomial time in $(\theta, \owedge, \mathcal{S})$ according to Propositions 3.
NP**-hardness:** We reduce the classical NP-complete problem SAT [2] to $[\ \owedge_{i=1}^{n}\ (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\mathrm{path}} \neq \emptyset$. An input of SAT is a set of clauses $\mathscr{C}$ over a set of propositions $\{p_1, \ldots p_r\}$, where each clause $C \in \mathscr{C}$ is a set of literals, that is either a proposition p_i or its negation $\neg p_i$. The SAT problem amounts to answering whether or not $\mathscr{C}$ is satisfiable, that is whether or not there is a valuation of the propositions $p_1, \ldots p_r$ that makes all clauses of $\mathscr{C}$ true. Now, let $\mathscr{C} = \{C_1, \ldots C_m\}$ over propositions $\{p_1, \ldots p_r\}$ be an input of the SAT problem; classically, $|\mathscr{C}| = \sum_{C \in \mathscr{C}} |C|$, where $|\mathscr{C}|$ is the number of literals that occur in C. We introduce two fresh propositions ι_0 and γ_0 and we define a labeled transition system $\mathcal{S}_{\mathscr{C}} = (S_{\mathscr{C}}, \rightarrow_{\mathscr{C}}, \lambda_{\mathscr{C}})$ over $\mathrm{Prop}_{\mathscr{C}} = \{\iota_0, \gamma_0, C_1, \ldots C_m\}$: In the following, we let ℓ_i denote either p_i or $\neg p_i$, for every $i \in \{1, \ldots, r\}$. We let $S_{\mathscr{C}} = \{s, t\} \cup \{\ell_i\}_{i=1,\ldots,r}$, $\rightarrow_{\mathscr{C}} = \{(\ell_i, \ell_{i+1}) \mid i \in [1; r-1]\} \cup \{(s, \ell_1), (\ell_r, t)\}$, and $\lambda_{\mathscr{C}} : \mathrm{Prop} \rightarrow 2^S$ is such that $\lambda_{\mathscr{C}}(\iota_0) = \{s\}$, $\lambda_{\mathscr{C}}(\gamma_0) = \{t\}$, and $\lambda_{\mathscr{C}}(\ell) = C$ whenever ℓ is a literal of C. An example of $\mathcal{S}_{\mathscr{C}}$ is depicted in Figure 4. Notice that by definition, $|\mathcal{S}_{\mathscr{C}}|$ is polynomial in $|\mathscr{C}|$.

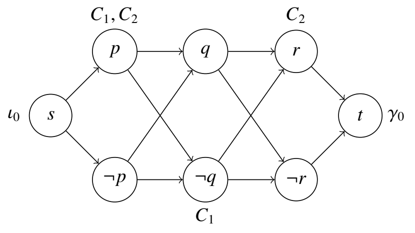


Fig. 4. $\mathcal{S}_{\mathscr{C}}$ with $\mathscr{C} = \{C_1, C_2\}$ where $C_1 = p \vee \neg q$ and $C_2 = p \vee r$

In the following, let call a *full path* a path of $\mathcal{S}_{\mathscr{C}}$ from s to t. The system $\mathcal{S}_{\mathscr{C}}$ is designed in such a way that any full path visits either p_j or $\neg p_j$ in an exclusive manner, for each $i = 1, \dots, r$. A full path π therefore unambigously denotes a valuation v_π of the propositions. Reciprocally, every valuation v of the propositions yields a unique full path π_v. Additionally, a full path π visits a state labeled by $C \in \mathscr{C}$ if, and only if, v_π makes the clause C true. Moreover, a full path π visits a state labeled by C if, and only if there is an anchoring $[0; j]$ such that $\pi(j)$ is labeled by C. The following concludes the proof of Proposition 5.

It remains to establish that $[(\iota_0 \blacktriangleright \gamma_0) \oslash (\iota_0 \blacktriangleright C_1) \oslash \dots \oslash (\iota_0 \blacktriangleright C_m)]^{\text{path}}_{\mathcal{S}_{\mathscr{C}}} \neq \emptyset$ if, and only if, $\mathscr{C}$ is satisfiable.

Assume $[(\iota_0 \blacktriangleright \gamma_0) \oslash (\iota_0 \blacktriangleright C_1) \oslash \dots \oslash (\iota_0 \blacktriangleright C_m)]^{\text{path}}_{\mathcal{S}_{\mathscr{C}}} \neq \emptyset$, with some element π. The constraint $(\iota_0 \blacktriangleright \gamma_0)$ enforces π to be a full path. Now, the other constraints enforce π to visits a state labeled by C for each $C \in \mathscr{C}$, entailing the valuation v_π making all clauses true, so that $\mathscr{C}$ is a positive input of SAT. It is not hard to see that conversely, if $\mathscr{C}$ is a positive instance of SAT, then any full path π_v, for some valuation v that makes all clauses true, is in $[(\iota_0 \blacktriangleright \gamma_0) \oslash (\iota_0 \blacktriangleright C_1) \oslash \dots \oslash (\iota_0 \blacktriangleright C_m)]^{\text{path}}_{\mathcal{S}_{\mathscr{C}}}$.

5.2 Complexity results for ADM(O), CONS(O), COMP(O)

We successively study admissibility, consistency, and completeness.

Theorem 1. ADM($\{\varovee, \ominus\}$) *is in* P.

Proof. We consider the syntactic fragment of the temporal logic CTL [1] defined by: $\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi$, where the only temporal operator is the "eventually" one denoted by $\Diamond$. The semantics of a formula φ of this fragment is given with regard to a labeled transition system $\mathcal{S} = (S, \rightarrow, \lambda)$, and is noted $[\![\varphi]\!]_{\mathcal{S}}$. We define $[\![\varphi]\!]_{\mathcal{S}} \subseteq S$ by induction over φ: $[\![p]\!]_{\mathcal{S}} = \lambda(p)$, $[\![\varphi \wedge \varphi']\!]_{\mathcal{S}} = [\![\varphi]\!]_{\mathcal{S}} \cap [\![\varphi']\!]_{\mathcal{S}}$, $[\![\varphi \vee \varphi']\!]_{\mathcal{S}} = [\![\varphi]\!]_{\mathcal{S}} \cup [\![\varphi']\!]_{\mathcal{S}}$, and $[\![\Diamond\varphi]\!]_{\mathcal{S}} = Pre^*_{\mathcal{S}}([\![\varphi]\!]_{\mathcal{S}})$, where $Pre^*_{\mathcal{S}}$ is defined in Section 3.1, that is $s \in [\![\Diamond\varphi]\!]_{\mathcal{S}}$ iff there is a path in $\mathcal{S}$ starting from s that visits a state in $[\![\varphi]\!]_{\mathcal{S}}$. In the following we simply write $[\![\varphi]\!]$ instead of $[\![\varphi]\!]_{\mathcal{S}}$. Note that computing $[\![\varphi]\!]$ takes polynomial time in the size of $\mathcal{S}$ and φ, see for example [9].

Let $\varphi^{\theta}_{\ovee} := \displaystyle\bigvee_{i=1}^{n} \iota \wedge \iota_i \wedge \Diamond(\gamma \wedge \gamma_i)$. Then $[\, \ovee_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}} \neq \emptyset$ if, and only if, $[\![\varphi^{\theta}_{\ovee}]\!] \neq \emptyset$. Indeed, if $[\, \ovee_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}} \neq \emptyset$, then let π be a path in $[\, \ovee_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}}$. Then π go from ι to γ and there is an $i \in [1; n]$ such that π go from ι_i to γ_i. Hence, π go from $\iota \wedge \iota_i$ to $\gamma \wedge \gamma_i$. So $\pi \in [\![\iota \wedge \iota_i \wedge \Diamond(\gamma \wedge \gamma_i)]\!]$, which implies $\pi \in [\![\varphi^{\theta}_{\ovee}]\!]$, which is then non-empty. Conversely, if $[\![\varphi^{\theta}_{\ovee}]\!] \neq \emptyset$, then let $\pi \in [\![\varphi^{\theta}_{\ovee}]\!]$. Then there is an $i \in [1; n]$ such that π start from ι_i and eventually reaches $\iota \wedge \gamma \wedge \gamma_i$. Let π' be the prefix of π that is a path that go from ι_i to $\iota \wedge \gamma \wedge \gamma_i$. We have $\pi' \in [\, \ovee_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}}$, so it is not empty.

Let $\varphi^{\theta}_{\ogreaterthan} := \iota \wedge \iota_1 \wedge \Diamond(\gamma_1 \wedge \iota_2 \wedge \Diamond(\gamma_2 \wedge \ldots \Diamond(\gamma_n \wedge \gamma)))$. Then $[\, \ogreaterthan_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}} \neq \emptyset$ if, and only if, $[\![\varphi^{\theta}_{\ogreaterthan}]\!] \neq \emptyset$ Indeed, if $[\, \ogreaterthan_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}} \neq \emptyset$, then let π be a path in $[\ogreaterthan_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}}$. Then π go from ι to γ and is the concatenation of paths π_i going from ι_i to γ_i. Hence, π visits successively $\iota \wedge \iota_1$, $\gamma_1 \wedge \iota_2$, …and $\gamma_n \wedge \gamma$. So $\pi \in [\![\varphi^{\theta}_{\ogreaterthan}]\!]$, which is then non-empty. Conversely, if $[\![\varphi^{\theta}_{\ogreaterthan}]\!] \neq \emptyset$, then let $\pi \in [\![\varphi^{\theta}_{\ogreaterthan}]\!]$. Then π visits successively $\iota \wedge \iota_1$, $\gamma_1 \wedge \iota_2$, …and $\gamma_n \wedge \gamma$. Let π' be the prefix of π that is a path that stops in $\gamma_n \wedge \gamma$. We have $\pi' \in [\, \ogreaterthan_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \cap [\iota \blacktriangleright \gamma]^{\text{path}}_{\mathcal{S}}$, so it is not empty.

Theorem 2. `ADM`$(\{\owedge\})$ *is* NP-*complete.*

Proof. First, we show that `ADM`$(\owedge)$ is NP-easy by giving a non-deterministic polynomial algorithm. According to Proposition 5, we can guess $\pi \in [\, \owedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}}$ with a non-deterministic polynomial algorithm. Then, we check that $\pi \in [(\iota \blacktriangleright \gamma)]^{\text{path}}_{\mathcal{S}}$ by checking that $\pi(0) \in \lambda(\iota)$ and

$\pi(|\pi|) \in \lambda(\gamma)$ which is done in constant time, so the algorithm is still polynomial.

Second, we prove the NP-hardness by reducing the problem of Proposition 5. Let $\mathcal{S} = (S, \to, \lambda)$ be a labeled transition system over propositions $\{\iota_1, \gamma_1, \ldots, \iota_n, \gamma_n\}$. We extend $\mathcal{S}$ to $\mathcal{S}' = (S, \to, \lambda')$ over propositions $\{\iota_1, \gamma_1, \ldots, \iota_n, \gamma_n\} \cup \{\iota, \gamma\}$, with $\lambda'(\iota) = \lambda'(\gamma) = S$ and λ' coincide with λ over $\{\iota_1, \gamma_1, \ldots, \iota_n, \gamma_n\}$. As $[(\iota \blacktriangleright \gamma)]^{\text{path}}_{\mathcal{S}'} = \Pi(\mathcal{S}')$, deciding $\mathtt{ADM}(\owedge)$ for $\mathcal{S}$ amounts to deciding if $[\ \owedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}'} \neq \emptyset$. So the reduction is straightforward, and $\mathtt{ADM}(\owedge)$ is NP-hard.

Theorem 3. $\mathtt{CONS}(\{\ovee\})$ *is in* P.

Proof. We show a polynomial algorithm deciding $\mathtt{CONS}(\{\ovee\})$.
Let $(\theta, \ovee, \mathcal{S})$ be an input of $\mathtt{CONS}(\{\ovee\})$. First, we compute the sets of state $S(\iota_i) = \lambda(\iota_i) \cap Pre^*_{\mathcal{S}}(\lambda(\gamma_i))$ and $S(\gamma_i) = \lambda(\gamma_i) \cap Post^*_{\mathcal{S}}(\lambda(\iota_i))$ for $i \in [1; n]$, which is done in polynomial time by reachability analysis. $S(\iota_i)$ (resp. $S(\gamma_i)$) represents the states of $\lambda(\iota_i)$ (resp. $\lambda(\gamma_i)$) from which there is a path to a state of $\lambda(\gamma_i)$ (resp. from a state of $\lambda(\iota_i)$). Then, check that for every $i \in [1; n]$, $S(\iota_i) \subseteq \lambda(\iota)$ and $S(\gamma_i) \subseteq \lambda(\gamma)$.

Theorem 4. $\mathtt{CONS}(\{\ogreaterthan\})$ *is in* co-NP.

Proof. We decribe a polynomial non-deterministic algorithm deciding that an input $(\theta, \ogreaterthan, \mathcal{S})$ of $\mathtt{CONS}(\{\ogreaterthan\})$ is a negative instance[9]. Let $(\theta, \ogreaterthan, \mathcal{S})$ be an input of $\mathtt{CONS}(\{\ogreaterthan\})$. We first guess a path π, and check that $\pi \in [\ \ogreaterthan_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}}$: the latter can be done by guessing anchoring intervals $[x_i; x_{i+1}]$ of a sequential decomposition of π, and by checking that for all $i \in [1; n]$, $x_i \leq x_{i+1}$, that $\pi(x_i) \in \lambda(\iota_i)$ and $\pi(x_{i+1}) \in \lambda(\gamma_i)$, and that $x_1 = 0$ and $x_{n+1} = |\pi|$. It then remains to verify that $\pi \notin [(\iota \blacktriangleright \gamma)]^{\text{path}}_{\mathcal{S}}$. Clearly all these checks can be done in polynomial time.

Theorem 5. $\mathtt{CONS}(\{\owedge\})$ *is* co-NP-*complete.*

Proof. A negative instance $(\theta, \owedge, \mathcal{S})$ of $\mathtt{CONS}(\{\owedge\})$ is characterized by the existence of a path $\pi \in [\ \owedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]^{\text{path}}_{\mathcal{S}} \setminus [(\iota \blacktriangleright \gamma)]^{\text{path}}_{\mathcal{S}}$.

[9] namely that the answer is "no".

Deciding that $(\theta, \varowedge, \mathcal{S})$ is a negative instance of $\texttt{CONS}(\{\varowedge\})$ is NP-easy: According to Proposition 5, we can guess $\pi \in [\ \varowedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}}$ with a non-deterministic polynomial-time algorithm. It then remains to check that $\pi \notin [(\iota \blacktriangleright \gamma)]_{\mathcal{S}}^{\text{path}}$ which takes constant time, by Proposition 1.
Deciding that $(\theta, \varowedge, \mathcal{S})$ is a negative instance of $\texttt{CONS}(\{\varowedge\})$ is NP-hard: we reduce the problem of deciding $[\ \varowedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}} \neq \emptyset$ (which is NP-complete by Proposition 5). Assume $\mathcal{S} = (S, \rightarrow, \lambda)$ is a labeled transition system over $\{\iota_1, \gamma_1, \ldots, \iota_n, \gamma_n\}$. We extend λ to the set of propositions $\{\iota_1, \gamma_1, \ldots, \iota_n, \gamma_n\} \cup \{\iota, \gamma\}$, by letting $\lambda(\iota) = \lambda(\gamma) = \emptyset$. By construction $[(\iota \blacktriangleright \gamma)]_{\mathcal{S}'}^{\text{path}} = \emptyset$, so that $(\theta, \varowedge, \mathcal{S})$ is a negative instance of $\texttt{CONS}(\{\varowedge\})$ is equivalent to $[\ \varowedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}'}^{\text{path}} \neq \emptyset$, and we are done.

We now turn to the decision problems $\texttt{COMP}(O)$. By Remark 1, a negative instance of $\texttt{CONS}(O)$ necessarily is a negative instance of $\texttt{COMP}(O)$.

Theorem 6. $\texttt{COMP}(\{\varovee, \ogreaterthan\})$ *is in* co-NP.

Proof. Let $(\theta, \Box, \mathcal{S})$ be a negative instance of $\texttt{COMP}(\{\varovee, \ogreaterthan\})$. There are two cases: (a) $(\theta, \Box, \mathcal{S})$ is a negative instance of $\texttt{CONS}(\Box)$, or (b) there exists a path $\pi \in [(\iota \blacktriangleright \gamma)]_{\mathcal{S}}^{\text{path}} \setminus [\ \Box_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}}$. The algorithm guesses whether it is Case (a) or Case (b). For Case (a) we use the polynomial-time algorithm in the proof of Theorem 3 (recall P $\subseteq$ co-NP) for operator $\varovee$ and Theorem 4 for operator $\ogreaterthan$. Regarding Case (b), we use a variant of the proof of Theorem 4: the non-deterministic polynomial-time algorithm that decides whether or not an input $(\theta, \Box, \mathcal{S})$ is a negative instance of $\texttt{COMP}(\{\varovee, \ogreaterthan\})$ consists in guessing a path π, and in checking that $\pi \in [(\iota \blacktriangleright \gamma)]_{\mathcal{S}}^{\text{path}} \setminus [\Box_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}}$. First, the algorithm guesses an elementary path π and checks $\pi \in [(\iota \blacktriangleright \gamma)]_{\mathcal{S}}^{\text{path}}$; this check is done in constant time by Proposition 1. Next, the algorithm checks that $\pi \notin [\ \Box_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}}$. If $\Box = \varovee$, we apply Proposition 1, otherwise $\Box = \ogreaterthan$ and we use Proposition 2.

The last theorem shows that the operator $\varowedge$ is much harder to handle.

Theorem 7. $\texttt{COMP}(\varowedge)$ *is in* Π_2^P.

Proof. We show that negative instances of $\texttt{COMP}(\varowedge)$ can be captured by a polynomial-time non-deterministic algorithm that can call a polynomial-time non-deterministic subroutine. Let $(\theta, \owedge, \mathcal{S})$ be a negative instance of `COMP`($\owedge$). Similarly to the case of Theorem 6, there are two possible cases: (a) $(\theta, \owedge, \mathcal{S})$ is a negative instance of `CONS`($\owedge$), or (b) there exists $\pi \in [(\iota \blacktriangleright \gamma)]_{\mathcal{S}}^{\text{path}} \setminus [\owedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}}$. Therefore, the algorithm first non-deterministically guesses if it is Case (a) or Case (b). For Case (a), it behaves like the polynomial-time non-deterministic algorithm proposed for `CONS`($\owedge$) in the proof of Theorem 5. Regarding Case (b), the algorithm guesses a path [10] π and checks that $\pi \in [(\iota \blacktriangleright \gamma)]_{\mathcal{S}}^{\text{path}}$. Then, the algorithm checks that $\pi \notin [\owedge_{i=1}^{n} (\iota_i \blacktriangleright \gamma_i)]_{\mathcal{S}}^{\text{path}}$. The latter can be performed by running an NP oracle according to Proposition 3. Hence the set of negative instances of `COMP`($\owedge$) is in $\mathrm{NP}^{\mathrm{NP}}$, that is Σ_2^P, which concludes.

6 Discussion

In this paper, we have developed a path semantics to attack trees that yields three natural notions of soundness of attack trees: *admissibility*, *consistency*, and *completeness*. Each soundness notion conveys a meaning of the practioners' manual decomposition of internal nodes. We then have explored the complexity of the associated decision problems.

	`ADM`	`CONS`	`COMP`
$\ovee$	P	P	co-NP
$\ogreaterthan$	P	co-NP	co-NP
$\owedge$	NP-complete	co-NP-complete	Π_2^P

Fig. 5. Complexities of the three decision problems for each operator

As can be seen, operators $\ovee$ and $\ogreaterthan$ are much simpler than the very classic operator $\owedge$ widely used in the literature; actually the complexity Π_2^P established here for the decision problem `COMP`($\owedge$) is not proved to be optimal, but we conjecture it is. This rather high complexity is not suprising since the notion of parallel decomposition underlying the operational semantics of operator $\owedge$ features complex combinatorics. As future work, we wish to complete the complexity classes picture by showing that, *e.g.* all complexities are tight.

[10] by Proposition 4 it is enough to consider paths whose size is polynomial in $(\theta, \owedge, \mathcal{S})$

References

1. Edmund M Clarke and E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, 1981.
2. Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 3–5 1971 1971.
3. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freeman, 1979.
4. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Rolando Trujillo-Rasua. Attack Trees with Sequential Conjunction. In *IFIP SEC 2015*, volume 455 of *IFIP AICT*, pages 339–353. Springer, 2015.
5. Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. Attack–Defense Trees. *Journal of Logic and Computation*, 24(1):55–87, 2014.
6. Gabriele Lenzini, Sjouke Mauw, and Samir Ouchani. Security analysis of socio-technical physical systems. *Computers & Electrical Engineering*, 47:258–274, 2015.
7. Sophie Pinchinat, Mathieu Acher, and Didier Vojtisek. Atsyra: An integrated environment for synthesizing attack trees - (tool paper). In *Graphical Models for Security - Second International Workshop, GraMSec 2015, Verona, Italy, July 13, 2015, Revised Selected Papers*, pages 97–101, 2015.
8. Bruce Schneier. Attack Trees. *Dr. Dobb's Journal of Software Tools*, 24(12):21–29, 1999.
9. Philippe Schnoebelen. The complexity of temporal logic model checking. *Advances in modal logic*, 4(393-436):35, 2002.
10. Larry J Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.