Guided design of attack trees: a system-based approach

Maxime Audinot, Sophie Pinchinat
Univ Rennes, CNRS, IRISA, Rennes, France
{maxime.audinot, sophie.pinchinat}@irisa.fr

Barbara Kordy
Univ Rennes, INSA Rennes, CNRS, IRISA, Rennes, France
barbara.kordy@irisa.fr

Abstract—**Attack trees are a well-recognized formalism for security modeling and analysis, but in this work we tackle a problem that has not yet been addressed by the security or formal methods community – namely guided design of attack trees. The objective of the framework presented in this paper is to support a security expert in the process of designing a pertinent attack tree for a given system. In contrast to most of existing approaches for attack trees, our framework contains an explicit model of the real system to be analyzed, formalized as a transition system that may contain quantitative information. The leaves of our attack trees are labeled with reachability goals in the transition system and the attack tree semantics is expressed in terms of traces of the system. The main novelty of the proposed framework is that we start with an attack tree which is not fully refined and by exhibiting paths in the system that are optimal with respect to the quantitative information, we are able to suggest to the security expert which parts of the tree contribute to optimal attacks and should therefore be developed further. Such useful parts of the tree are determined by solving a satisfiability problem in propositional logic.**

I. Introduction

Attack trees [26], [21] are probably the best known and the most often used in practice modeling language for security and risk analysis [10], [24], [18]. The first strong point of attack trees is that they provide a simple and very intuitive way of representing and structuring attacks. The objective of the attacker is depicted with the root node of the tree and the remaining nodes refine this objective into smaller and easier to grasp subgoals that need to be reached to achieve the root goal. This hierarchical structure of attack trees implies the second great advantage of the model, namely the possibility of performing efficient quantitative evaluation of security of the analyzed system. By estimating time, cost, impact, or probability of various attacks, the optimal (from the point of view of the attacker) attack paths can be found. This allows the analyst to identify the corresponding vulnerabilities, and thus identify the weakest points in the system, where patches should be applied or detection mechanisms should be put in place.

The setting: In this work, we consider attack trees with `OR`, `AND`, and `SAND` refinements. We follow a system-based approach, close to the one recently proposed in [4], where the real system to be analyzed with the help of an attack tree is modeled explicitly as a transition system. Furthermore, the labels of the leaf nodes in our attack trees represent reachability goals in this system and can be more complex or more general than classical basic actions used to label the leaves in the standard attack tree approaches. We can thus consider trees that are not yet fully deployed, i.e., whose leaves are to be further refined. The semantics of our attack trees relies on the notion of traces that can be interpreted in the underlying system. Next, by a monitoring technique, we can focus on all attacks captured by our attack tree formalization that also exist in the analyzed real system. This way, we can capture more dependencies than those represented by the `AND` and `SAND` attack tree refinements. The quantitative values used in our approach originate from the system. This implies that the value of executing an action (that intuitively may correspond to taking a weighted transition in the transition system) may depend on the context of this action in the system. In other words, executing the same action may take different values in different places or points in time in the system.

Objective: This work is concerned with the problem of guiding a security expert in the process of designing a relevant attack tree. By relevant, we understand a tree where only the potentially useful parts (i.e., parts that may contribute to an optimal attack) are developed, and where the constraints originating from the analyzed system are taken into account in the quantitative analysis.

- We start with an attack tree τ that is not fully deployed (potentially composed of the root node only). The labels of the leaves of this tree represent the reachability goals in the transition system $\mathcal{S}$ modeling the analyzed system.
- We construct an automaton $\mathcal{A}_\tau$ that accepts the trace semantics of our attack tree, independently of the system.
- We compute a product $\tau[\mathcal{S}]$ between the transition system and the attack tree automaton to restrict the system behavior to the situations modeled by the attack tree only, as a kind of monitoring.
- By a quantitative analysis of $\tau[\mathcal{S}]$, typically by solving optimal reachability problems, we can synthesize traces involved in the optimal attacks, called *witnesses*.
- Given a witness, we provide a method to identify the leaves in τ that may be involved in such an optimal attack. We call them *useful leaves*. We reduce the problem of determining if a leaf is useful to a propositional satisfiability problem.
- Identifying useful leaves allows us to suggest to the security expert which parts of his tree he should refine and which are unnecessary for further analysis.

Using the above recipe, the security expert is advised on which parts of a partially deployed attack tree contribute to the optimal attacks and should therefore be further refined. As a consequence, the tree constructed using our guided approach would be smaller than a totally deployed tree built manually, in a standard, brainstorming-based way. The guidance does not prevent the expert from refining other nodes, if he wishes, but tells him they may be less important w.r.t. the quantitative analysis he is interested in.

The paper is organized as follows. Related work is described in Section II. We devote Section III to the introduction of new operations on formal languages that will be used to formalize the semantics of our attack trees in Section IV where the construction of the automaton $\mathcal{A}_\tau$ for attack tree τ is provided. In Section V we explain how to monitor a transition system with the automaton $\mathcal{A}_\tau$, and how to perform early-stage quantitative analysis in this context. Finally, in Section VI, we exploit the notion of useful positions to formalize our method for guided design of attack trees, summarized in Section VI-D. We conclude in Section VII.

II. Related work

Classical formalizations of attack trees, cf., [21], [15], [14], are usually based on the notion of basic actions: leaves of attack trees are labeled with actions to be executed by the attacker and the intermediate nodes describe how these actions should be combined (using `OR`, `AND`, and sometimes `SAND` refinements) to construct an attack. A naive, but standard, quantitative analysis of such trees relies on an incremental bottom-up approach, intuitively introduced by Schneier in [26] and formalized for the first time by Mauw and Oostdijk in [21], where values are assigned to the basic actions and then propagated to the remaining nodes, depending on the type of the node's refinement. One of the weaknesses of this approach is that the dependencies between the actions cannot be expressed, and are thus not taken into account in the value propagation. In addition, since the analyzed system is not modeled explicitly, the optimal value obtained from the bottom-up analysis may in reality not be obtainable in the underlying system. Thus, in reality the bottom-up evaluation often gives an approximated result rather than the exact, optimal value.

This observation attracted the attention of researchers with the formal methods background, who proposed analysis methods for attack trees based on operational semantics, e.g., [19], [11]. Such methods permit to better capture the correlations between actions and benefit from well-known verification methods, such as optimal reachability, model checking with temporal logic, etc. In this context, Kumar et al. [19] pioneered the use of automata for the analysis of attack trees. They proposed an effective way of computing the necessary resources, e.g., time, skills, etc., and the corresponding attack paths achieving the root goal of an attack tree. Their method also allows to rank possible attack paths according to a given quantitative criterion, for instance identify ten cheapest attack paths. To do so, the leaves of an attack tree are augmented with a cost structure capturing several quantitative components, e.g., time, skills, resources, that can be dependent on each other. The leaves of the tree are then translated into priced timed automata (PTA) [7], [2] expressing how the corresponding basic actions behave in the analyzed system. These basic automata are combined using PTAs for `OR`, `AND`, and `SAND` refinements to provide the semantics of the intermediate nodes. The resulting network of automata is model checked using the Uppaal CORA tool [29] against the quantitative queries expressed in weighted CTL.

In [11], Gadyatskaya et al. propose an alternative way of using timed automata to analyze attack–defense trees [17] – an extension of classical attack trees with countermeasure nodes. The main novelty of [11] is to separate the modeling of the attacker's and the defender's behavior from the modeling of the security scenario itself. In this work, the attacker, the defender, and the environment are encoded as timed automata communicating with each other. The modeling of the environment contains automata for basic actions related to the elements composing the system, such as locations, assets, etc. The attacker is stochastic (he selects his actions probabilistically) and constraint by time and available resources. The attacker's actions are decorated with an interval expressing their execution time, a probability mass, and the cost. The structure of an attack–defense tree is encoded as a Boolean formula that the attacker wishes to become true while taking quantitative and probabilistic constraints into account. The Uppaal tool is employed to, given a set of countermeasures deployed by the defender, find an attack that succeeds within a specified time interval, estimate the probability of a successful attack within a given time-bound, or compute the corresponding estimated cost for the attacker.

The common elements of the works presented above are the following

1) Modeling of the system is reduced to modeling the underlying basic actions.
2) The quantitative input values are assigned to the actions represented by the leaves of the tree.
3) The objective of standard approaches is to analyze an existing attack tree and not to update or modify it.

These three aspects constitute the main differences between the existing approaches and the framework that we develop in this paper. 1) We model the analyzed system explicitly, as a transition system, and a link is made (language-wise) between the system states and the labels of the attack tree nodes. More precisely, the labels in our attack trees describe reachability goals in the system model. This way the link between the attack tree nodes and the system behavior is mathematically robust, at every step of the tree deployment. Thus, our goal-labeled attack trees are more precise than those labeled with basic actions, because their labels correspond exactly to what can happen in the analyzed system. In addition, our method takes the dependencies that are inherent in the system model into account. Our product construction between the tree (automaton) and the system gives a direct mean to enforce these dependencies. 2) The formal model of the system to be analyzed may contain quantitative information that is used to find optimal attacks. Such an approach is more precise than the existing approaches relying on values assigned to basic actions, because it allows for differentiating between the values of similar actions performed in different contexts. As a result, our quantitative evaluation provides more accurate and more realistic results compared to the existing approaches where the values are assigned to basic actions and where the system to be analyzed is not modeled explicitly. 3) The ultimate goal of this work is to provide a formal approach to support the security expert in the process of developing the tree. To the best of our knowledge, we are the first ones to have the objective of guiding the tree design with respect to the analyzed system and the underlying quantitative information.

III. New operations on formal languages

Given a set X of propositions, we let $C^+(X)$ be the set of positive conjunctive Boolean formulas over X and we refer to $\top$ for the true formula, the neutral element of operator $\wedge$. The elements of $C^+(X)$ will be denoted by $b, b', b_i \ldots \in C^+(X)$, Given two formulas $b, b' \in C^+(X)$, we write $b \models b'$ whenever b' is a logical consequence of b; in particular, for every $b \in C^+(X)$, we have $b \models \top$.

In our setting, the set $C^+(X)$ will be seen as an alphabet, and we use notations $u_1, u_2, \ldots$ for finite words over this alphabet, i.e., the elements of $C^+(X)^*$. The symbol ϵ represents the empty word. The *size* of a word $u \in C^+(X)^*$, written $|u|$, is the length of the sequence u; thus $|\epsilon| = 0$. For any $1 \leq i \leq |u|$, the expression $u(i)$ denotes the i-th letter of word u. Also, for any $1 \leq i \leq j \leq |u|$, the expression $u[i, j]$ is the subword of u starting at $u(i)$ and ending at $u(j)$. We also use the standard notation for concatenation of words u and v: namely, $u.v$, or simply uv.

A. Enhanced concatenation and shuffle

We now define two binary operations over (sets of) words: the *enhanced concatenation* (Definition 1) and the *shuffle*[1] (Definition 3). Intuitively, the objective of the enhanced concatenation will be to capture the sequential composition of attacker's goals: either the goals are achieved one after the other (standard concatenation), or the goals are achieved simultaneously which is reflected by a logical conjunction. The shuffle captures the achievements of several goals: it is inherently combinatorial because the order of these achievements may be arbitrary, and as for the enhanced concatenation, some goals may be achieved simultaneously.

Definition 1 (Enhanced concatenation of words)**.** *Given two words $u_1, u_2 \in C^+(X)^*$ and two letters $b_1, b_2 \in C^+(X)$, we let the* enhanced concatenation *of u_1 and u_2 be the subset $u_1 \odot u_2 \subseteq C^+(X)^*$ defined by induction over words as follows*

- $u \odot \epsilon = \epsilon \odot u = \{u\}$, *for all words* $u \in C^+(X)^*$,
- $(u_1 b_1) \odot (b_2 u_2) = \{u_1 b_1 b_2 u_2, u_1(b_1 \wedge b_2)u_2\}$

[1]There are many other notions of shuffle in the literature, see e.g., [9].

For example, $b \odot \top b' = \{b \top b', bb'\}$, where we freely simplify all expressions of the form $b \wedge \top$ or $\top \wedge b$ by b. We now can generalize the enhanced concatenation to languages as follows.

Definition 2 (Enhanced concatenation of languages)**.** *Let L and L' be two languages over alphabet $C^+(X)$. We define the* enhanced concatenation *of L and L' as*

$$L \odot L' = \bigcup_{u \in L,\ u' \in L'} u \odot u'.$$

Definition 3 of the shuffle operation is inspired by, e.g. [13].

Definition 3 (Shuffle of words)**.** *Given two words $u_1, u_2 \in C^+(X)^*$ and two letters $b_1, b_2 \in C^+(X)$, we let the* shuffle *of u_1 and u_2 be the subset $u_1 \otimes u_2 \subseteq C^+(X)^*$ defined by induction over words as follows*

- $u \otimes \epsilon = \epsilon \otimes u = \{u\}$, *for all words* $u \in C^+(X)^*$,
- $b_1 u_1 \otimes b_2 u_2 = b_1(u_1 \otimes b_2 u_2) \cup b_2(b_1 u_1 \otimes u_2) \cup (b_1 \wedge b_2)(u_1 \otimes u_2)$

For example, $b_1 b_2 \otimes b_3 = \{b_1 b_2 b_3, b_1(b_2 \wedge b_3), b_1 b_3 b_2, (b_1 \wedge b_3) b_2, b_3 b_1 b_2\}$. The shuffle operation is generalized on languages as follows.

Definition 4 (Shuffle of languages)**.** *Let $L, L' \subseteq C^+(X)^*$ be two languages. We define the* shuffle *of L and L' as*

$$L \otimes L' = \bigcup_{u \in L,\ u' \in L'} u \otimes u'.$$

We end this section with few notions that will be used later.

Definition 5. *A word v is* embedded *into a word u, written $v \models u$, if $|v| = |u|$, and for every $1 \leq i \leq |v|$, $v(i) \models u(i)$.*

For example, $b_1.(b_1 \wedge b_2).(b_1 \wedge b_2) \models \top.b_2.(b_1 \wedge b_2)$.

We next show that enhanced concatenation and shuffle preserve the regularity of languages by providing automata constructions for each of them.

B. Automata for enhanced concatenation and shuffle

The property that the class of regular languages is closed by some operation, for instance complementation, often relies on finite-state automata constructions (see [12, Chaper 3, Section 3.2]).

Definition 6. *A* finite-state automaton *over a finite alphabet A is a tuple $\mathcal{A} = (Q, A, \delta, I, F)$, where Q is a set of* states*, $\delta : Q \times A \to 2^Q$ is the* transition function[2]*, $I \subseteq Q$ is the set of* initial states*, and $F \subseteq Q$ is the set of* final states*, that w.l.o.g. we may assume "terminal"[3]. The* language *of $\mathcal{A}$, written $L(\mathcal{A})$, is the set of words $a_0 \ldots a_n \in A^*$ for which there exists a sequence of states of $\mathcal{A}$ of the form $q_0, \ldots, q_{n+1} \in Q$, with $q_0 \in I$, $q_{n+1} \in F$, and $q_i \in \delta(q_{i-1}, a_{i-1})$, for $1 \leq i \leq n+1$. The sequence $q_0, \ldots, q_{n+1}$ is called a* run *of $\mathcal{A}$, and since $q_{n+1} \in F$ it is also* accepting *the word $a_0 \ldots a_n$.*

We make use of automata to obtain the results stated in Proposition 1.

[2]We consider non-deterministic automata.

[3]A terminal state does not have any outgoing transitions.

Proposition 1. *The class of regular languages is closed by enhanced concatenation and by shuffle.*

The rest of this section is dedicated to the proof of Proposition 1.

Let us fix two automata $\mathcal{A}_1 = (Q_1, C^+(X), \delta_1, q_1^0, F_1)$ and $\mathcal{A}_2 = (Q_2, C^+(X), \delta_2, q_2^0, F_2)$. We will successively define automata $\mathcal{A}_1 \odot \mathcal{A}_2$ and $\mathcal{A}_1 \otimes \mathcal{A}_2$ in such a way that Proposition 2 holds (its full proof can be found in Appendix A). Proposition 1 is then a mere corollary.

Proposition 2. $L(\mathcal{A}_1 \odot \mathcal{A}_2) = L(\mathcal{A}_1) \odot L(\mathcal{A}_2)$ *and* $L(\mathcal{A}_1 \otimes \mathcal{A}_2) = L(\mathcal{A}_1) \otimes L(\mathcal{A}_2)$.

The construction for the enhanced concatenation $\odot$ is inspired from the standard concatenation. The main difference reflects the ability to concatenate two words by merging the last letter of the former with the first letter of the latter. We define $\mathcal{A}_1 \odot \mathcal{A}_2 = (Q_1 \uplus Q_2, C^+(X), \delta, q_1^0, F_2)$ (where $\uplus$ denotes the disjoint union) by letting the transition function $\delta : (Q_1 \uplus Q_2) \times C^+(X) \to 2^{Q_1 \uplus Q_2}$ be such that $q' \in \delta(q, b)$ whenever one of the four following cases holds

- $q \in Q_1$ and $q' \in \delta_1(q, b)$,
- $q \in Q_2$ and $q' \in \delta_2(q, b)$,
- $q \in F_1$ and $q' \in \delta_2(q_2^0, b)$,
- letter b is of the form $b_1 \wedge b_2$ such that there exists $q_1 \in \delta_1(q, b_1) \cap F_1$, and $q' \in \delta_2(q_2^0, b_2)$.

For the shuffle operator $\otimes$, we define $\mathcal{A}_1 \otimes \mathcal{A}_2 = (Q_1 \times Q_2, C^+(X), \delta, (q_1^0, q_2^0), F_1 \times F_2)$ by letting the transition function $\delta : Q_1 \times Q_2 \times C^+(X) \to 2^{Q_1 \times Q_2}$ be such that: $(q_1', q_2') \in \delta((q_1, q_2), b)$ whenever one of the three following cases holds

- $q_1' = q_1$ and $q_2' \in \delta_2(q_2, b)$,
- $q_1' \in \delta_1(q_1, b)$ and $q_2' = q_2$,
- letter b is of the form $b_1 \wedge b_2$ with $q_1' \in \delta_1(q_1, b_1)$ and $q_2' \in \delta_2(q_2, b_2)$.

Now that enhanced concatenation and shuffle can be reflected by operations over automata, we can enter the core sections of our contribution which starts with a description of the attack tree framework we consider in this work.

IV. The attack tree framework

We first introduce the formal framework of attack trees, next their trace semantics and related automata constructions.

A. Attack trees

In the spirit of [4], attack trees are finite labeled trees whose leaves are labeled by *propositions* ranging over a fixed set Prop, and whose internal nodes (non leaves) are labeled by a (binary) *operator* ranging over $O = \{\texttt{OR}, \texttt{SAND}, \texttt{AND}\}$. The elements of O are respectively called “Or”, “Sequential And” and “And”. However, as opposed to [4] internal nodes do not display any labeling with propositions as these are not relevant in this framework. The meaning of operators OR, SAND, and AND is provided by the *trace semantics* of attack trees (see Definition 10).

Before introducing these notions formally, we briefly recall the mathematical setting for labeled binary trees. Rather than “node” we will use “position”. Let Σ be a set of labels.

Definition 7 (Σ-labeled binary trees)**.** *A* Σ-labeled (binary) tree *is a partial function* $\tau : \{1,2\}^* \to \Sigma$ *whose domain* $Pos(\tau)$*, called the set of* positions *of* τ*, is a finite, prefix-closed subset of* $\{1,2\}^*$ *such that for every position* $p \in Pos(\tau)$*, if* $p.2 \in Pos(\tau)$ *then* $p.1 \in Pos(\tau)$*.*

We write $p \leq q$ *whenever* p *is a prefix-word of* q *(*p *is the position of an ancestor node of* q*).*

For any position of the form $p.i$ *we let* $par(p.i) = p$ *and we call it the* parent position *of* $p.i$*.*

A Σ-labeled tree τ is a *leaf* whenever $\text{Pos}(\tau) = \{\epsilon\}$. Given a position $p \in \text{Pos}(\tau)$, the *subtree of* τ *at position* p is the Σ-labeled tree $\tau_{|p}$ defined by $\text{Pos}(\tau_{|p}) := \{p' \mid pp' \in \text{Pos}(\tau)\}$ and $\tau_{|p}(p') = \tau(pp')$. In the sequel, we will identify a leaf τ where $\tau(\epsilon) = \sigma$, with its label σ. We write $\sigma(\tau_1, \tau_2)$ for the Σ-labeled tree τ defined by $\tau(\epsilon) = \sigma \in \Sigma$, $\tau_1 = \tau_{|1}$, and $\tau_2 = \tau_{|2}$; the trees τ_1 and τ_2 are the *immediate subtrees* of τ.

Definition 8. *Let* τ *be a tree and let* $p \in Pos(\tau)$*. For any tree* τ'*, the* substitution of τ' in τ at position p *is the tree* $\tau[p \leftarrow \tau']$ *defined by replacing in* τ *the subtree* $\tau_{|p}$ *by the tree* τ'*.*

We now turn to the framework of attack trees where the set of labels is specialized to $\text{Prop} \cup O$ (recall that Prop is a set of propositions and $O = \{\texttt{OR}, \texttt{SAND}, \texttt{AND}\}$).

Definition 9 (Attack tree)**.** *The class of* attack trees *over* Prop *is the least subset of* (Prop$\cup O$)*-labeled trees such that for every* $\gamma \in \text{Prop}$*,* γ *is an attack tree, and if* τ_1 *and* τ_2 *are attack trees and* $\texttt{OP} \in O$*, then* $\texttt{OP}(\tau_1, \tau_2)$ *is also an attack tree.*

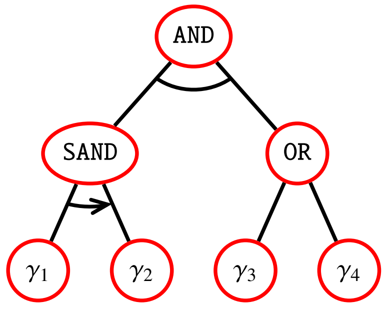

Fig. 1. The attack tree τ^e, with the graphical conventions of ADTool [16]

For reasons that will become clear in the next section, we will refer to propositions of Prop appearing in an attack tree as *goals*. Also, a position labeled by $\texttt{OP} \in O$ is called an OP position (and it is not a leaf position).

Given an attack tree τ over Prop, we define $\ell(\tau), \text{Pos}_{\texttt{OR}}(\tau)$, $\text{Pos}_{\texttt{SAND}}(\tau)$, $\text{Pos}_{\texttt{AND}}(\tau)$ as the set of leaf positions, the set of OR positions, the set of SAND positions, and the set of AND positions in τ, respectively. Formally,

- $\ell(\tau) = \bigcup_{\gamma \in \text{Prop}} \{p \in \text{Pos}(\tau) \mid \tau(p) = \gamma\}$;
- for $\texttt{OP} \in O$, $\text{Pos}_{\texttt{OP}}(\tau) = \{p \in \text{Pos}(\tau) \mid \tau(p) = \texttt{OP}\}$.

Example 1. *Let* τ^e *be the attack tree of Figure 1. Its set of positions is* $Pos(\tau^e) = \{\epsilon, 1, 2, 1.1, 1.2, 2.1, 2.2\}$*. Its root label is* $\tau^e(\epsilon) = \texttt{AND}$*. Its two immediate subtrees* ${\tau^e}_{|1}$ *and* ${\tau^e}_{|2}$

are respectively $\mathtt{SAND}(\gamma_1, \gamma_2)$ *and* $\mathtt{OR}(\gamma_3, \gamma_4)$. *The set of leaf positions of* τ^e *is* $\ell(\tau^e) = \{1.1, 1.2, 2.1, 2.2\}$, *where* $\tau^e(1.1) = \gamma_1$, $\tau^e(1.2) = \gamma_2$, $\tau^e(2.1) = \gamma_3$, $\tau^e(2.2) = \gamma_4$. *Also,* $Pos_{\mathtt{OR}}(\tau^e) = \{2\}$, $Pos_{\mathtt{SAND}}(\tau^e) = \{1\}$, $Pos_{\mathtt{AND}}(\tau^e) = \{\epsilon\}$.

B. Trace semantics and automata for attack trees

Attack trees can be given a *trace semantics* defined as a language over the alphabet $C^+(\mathrm{Prop})$.

Definition 10 (Trace semantics)**.** *Let* τ *be an attack tree over* Prop. *The* trace semantics *of* τ *is* $L(\tau) \subseteq C^+(\mathrm{Prop})^*$ *defined by induction over the structure of* τ *as follows.*

- $L(\gamma) = \top^*.\gamma$;
- $L(\mathtt{OR}(\tau_1, \tau_2)) = L(\tau_1) \cup L(\tau_2)$;
- $L(\mathtt{SAND}(\tau_1, \tau_2)) = L(\tau_1) \odot L(\tau_2)$;
- $L(\mathtt{AND}(\tau_1, \tau_2)) = L(\tau_1) \otimes L(\tau_2)$.

Elements of $L(\tau)$ *are called the* traces *of* τ.

Example 2. *Let* τ^e *be the attack tree from Figure 1. The trace semantics of* ${\tau^e}_{|1}$ *is* $L({\tau^e}_{|1}) = \top^*.\gamma_1 \odot \top^*.\gamma_2$, *which is* $\top^*.(\gamma_1 \wedge \gamma_2) \cup \top^*.\gamma_1.\top^*.\gamma_2$. *Also,* $L({\tau^e}_{|2}) = \top^*.\gamma_3 \cup \top^*.\gamma_4$. *Therefore* $L(\mathtt{AND}({\tau^e}_{|1}, {\tau^e}_{|2})) = L({\tau^e}_{|1}) \otimes L({\tau^e}_{|2})$, *which is complex to describe, so we only pick some of its elements: for example,* $\top^*.\gamma_1.\top^*.\gamma_2.\top^*.\gamma_3$, *and* $\top^*.\gamma_3.\top^*.\gamma_1.\top^*.\gamma_2$, *and* $\top^*.\gamma_1.\top^*.\gamma_3.\top^*.\gamma_2$, *and* $\top^*.(\gamma_1 \wedge \gamma_2 \wedge \gamma_3)$, *and* $\top^*.\gamma_1.\top^*.(\gamma_2 \wedge \gamma_4)$.

We now establish that the trace semantics of an attack tree is a regular language. This is achieved by providing finite-state automata constructions, which not surprisingly because of Definition 10, use operators $\odot$ and $\otimes$ between automata (see Section III). The result of Theorem 1 is not trivial because the trace semantics of an attack tree is not a finite language, but follows from Proposition 2 as the non-standard operations of enhanced concatenation and shuffle do not yield non-regularity.

Theorem 1. *Let* τ *be an attack tree over* Prop. *Then,* $L(\tau)$ *is a regular language. Moreover, one can effectively construct the* trace automaton *of* τ, *written* $\mathcal{A}_\tau$, *that is a finite-state automaton over* $C^+(\mathrm{Prop})$ *with* $L(\mathcal{A}_\tau) = L(\tau)$.

The rest of this section is dedicated to proving Theorem 1.

Notice that the ability to characterize $L(\tau)$ by means of a finite-state automaton $\mathcal{A}_\tau$ entails the regularity of $L(\tau)$. The construction of $\mathcal{A}_\tau$ is inductive over τ.

- For a leaf attack tree γ, by Definition 10 we have $L(\gamma) = \top^*.\gamma$ which is clearly regular. A natural automaton that accepts the language $\top^*.\gamma$ is depicted in Figure 2, where q is the unique final state.

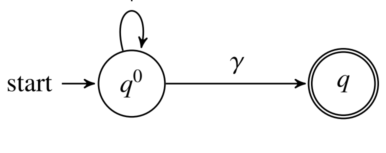

Fig. 2. The trace automaton $\mathcal{A}_\gamma$.

- For an attack tree τ of the form $\mathtt{OP}(\tau_1, \tau_2)$, by induction hypothesis, there exists some automata $\mathcal{A}_{\tau_1}$ and $\mathcal{A}_{\tau_2}$ that accept $L(\tau_1)$ and $L(\tau_2)$ respectively. We now consider the three possible cases for OP:
  - If $\mathtt{OP} = \mathtt{OR}$, we define $\mathcal{A}_{\mathtt{OR}(\tau_1,\tau_2)} = \mathcal{A}_{\tau_1} \cup \mathcal{A}_{\tau_2}$ that trivially accepts $L(\mathtt{OR}(\tau_1, \tau_2))$.
  - If $\mathtt{OP} = \mathtt{SAND}$, we let $\mathcal{A}_{\mathtt{SAND}(\tau_1,\tau_2)} = \mathcal{A}_{\tau_1} \odot \mathcal{A}_{\tau_2}$. By Proposition 2, $L(\mathcal{A}_{\mathtt{SAND}(\tau_1,\tau_2)}) = L(\tau_1) \odot L(\tau_2)$, which by definition is equal to $L(\mathtt{SAND}(\tau_1, \tau_2))$.
  - If $\mathtt{OP} = \mathtt{AND}$, similarly to the previous case we let $\mathcal{A}_{\mathtt{AND}(\tau_1,\tau_2)} = \mathcal{A}_1 \otimes \mathcal{A}_2$.

This last case achieves the proof of Theorem 1.

The construction of $\mathcal{A}_\tau$ may be used to verify the *membership problem*, that is given a tree τ and a word u, do we have $u \in L(\tau)$? Because the automaton in the worse case (AND-nodes only) is of size exponential in $|\tau|$ (recall that by construction $\left|\mathcal{A}_{\mathtt{AND}(\tau_1,\tau_2)}\right| = \left|\mathcal{A}_{\tau_1}\right| \times \left|\mathcal{A}_{\tau_2}\right|$), and because the emptiness of an automaton can be decided in NLOGSPACE [28], the membership problem can be solved in polynomial time.

Most importantly, the trace automaton $\mathcal{A}_\tau$ can be used to perform quantitative analysis. Before getting to this, we explain how the trace automaton of an attack tree can monitor a system.

V. System-based quantitative analysis

We consider models of systems that rely on standard transition systems and explain how to monitor the latter with the trace automaton of an attack tree. Then, we use this monitoring technique to reduce the synthesis problem of an optimal attack to the standard optimal reachability problem in *quantitative models*[4].

In the rest of this section, we fix an attack tree τ over some set Prop. W.l.o.g. we may assume that $\mathrm{Prop} = \{\gamma \,|\, \gamma \text{ is a leaf of } \tau\}$.

A. Transition systems

Transition systems are very common models, based on states and transitions between states, that naturally describe the dynamics of a system.

Definition 11. *An* Act-labeled transition system *over* Prop *is a tuple* $\mathcal{S} = (S, S^0, Act, \rightarrow, \lambda)$, *where*

- S *is a finite set of* states *(elements of* S *are denoted by* $s, s', s_0, s_1, \ldots$*), and* $S^0 \subseteq S$ *is the set of* initial *states,*
- Act *is a set of* actions, *whose typical element is* a,
- $\rightarrow \subseteq S \times Act \times S$ *is the* transition relation, *and we write* $s \overset{a}{\rightarrow} s'$ *instead of* $(s, a, s') \in \rightarrow$, *and* $s \rightarrow s'$ *whenever there is some action* a *such that* $s \overset{a}{\rightarrow} s'$,
- $\lambda : S \rightarrow 2^{\mathrm{Prop}}$ *is the* valuation *function that assigns propositions to states.*

Paths of transition systems are central objects as they represent system executions.

[4]Extensions of transition systems that own quantitative features, such as priced timed systems [7], [2].

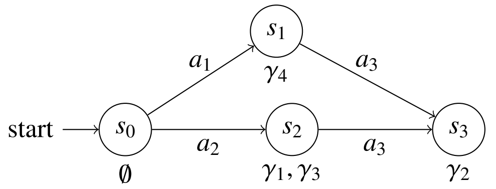

Fig. 3. The transition system $\mathcal{S}^{\mathrm{e}}$.

Definition 12. *A* path *of $\mathcal{S}$ is a sequence of states $\pi = s_0 \dots s_n$, such that $s_i \to s_{i+1}$, for every $0 \le i \le n-1$. The set of paths of $\mathcal{S}$ is denoted with $\Pi(\mathcal{S})$. An* initial path *is a path starting from an initial state of $\mathcal{S}$.*

Definition 13 (Traces)**.** *Given a path $\pi = s_0 s_1 \dots s_n \in \Pi(\mathcal{S})$, we let the* trace *of π be the finite word over alphabet 2^{Prop} defined by* $\mathrm{trace}(\pi) := \lambda(s_0)\lambda(s_1)\dots\lambda(s_n)$.

Remark 1 (Important)**.** *Notice that a trace $\lambda(s_0)\lambda(s_1)\dots\lambda(s_n)$ can be interpreted as a finite word over $C^+(\mathrm{Prop})$ by replacing each $\lambda(s_i)$ by the Boolean formula $\bigwedge_{\gamma\in\lambda(s_i)}\gamma$, with the convention that $\bigwedge_{\gamma\in\emptyset}\gamma = \top$ (the true formula). It therefore makes sense to state that a trace of $\mathcal{S}$ is embedded into some word of $L(\tau)$ (see Definition 5).*

We say that $\pi \in \Pi(\mathcal{S})$ is a *τ-attack on $\mathcal{S}$*, if π is an initial path and $\mathrm{trace}(\pi)$ can be embedded into some $u \in L(\tau)$. We abuse notation by writing $\mathrm{trace}(\pi) \in L(\tau)$ to mean that π is a τ-attack.

Example 3. *Let $\mathcal{S}^e$ be the transition system over* Prop $= \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$ *of Figure 3 (an arrow with "start" is meant to distinguish initial states, here s_0). This system has four states s_0, s_1, s_2, s_3 and three actions a_1, a_2, a_3. The labels of the states are $\lambda(s_0) = \emptyset$, $\lambda(s_1) = \{\gamma_4\}$, $\lambda(s_2) = \{\gamma_1, \gamma_3\}$, $\lambda(s_3) = \{\gamma_2\}$. An example of an initial path of $\mathcal{S}^e$ is $\pi^e = s_0 s_2 s_3$, whose trace is* $\mathrm{trace}(\pi^e) = \emptyset\{\gamma_1, \gamma_3\}\{\gamma_2\}$. *As stated in Remark 1, we can see this trace as the word $u^e = \top(\gamma_1 \wedge \gamma_3)\gamma_2 \in C^+(\mathrm{Prop})$. Incidentally, $u^e \in L(\tau^e)$ (see Figure 1), so that π^e is a τ^e-attack on $\mathcal{S}^e$. Also the path $\pi'^e = s_0 s_1 s_3$ has trace $v^e = \top\gamma_4\gamma_2$ and one can easily verify that $v^e \notin L(\tau^e)$.*

The concept of transition system is central to any sort of operational models and in particular to the ones that fall into classes of quantitative systems, such as timed systems [3], weighted automata [1], and the rich setting of priced timed automata [7], [2] (a standard input to the Uppaal CORA tool [29]). As our methodology is meant to be general, we abstract from the kind of quantitative systems one is interested in by agreeing on a notation: a *quantitative system* (or simply a *system*) is a pair $(\mathcal{S}, Q)$ where $\mathcal{S}$ is the underlying transition system and Q is some extra information that gathers the quantitative features, if any.

We can exploit the trace automaton $\mathcal{A}_\tau$ to restrict the system's behavior to τ-attacks by performing some product, called the *τ-monitoring of the system.*

B. System monitored by an attack tree

To monitor the system, we use a product (Definition 14) that generalizes the standard product construction between a transition system $\mathcal{S}$ and a finite-state/Büchi automaton $\mathcal{A}$, that we write $\mathcal{S}\times\mathcal{A}$, and used to model-check linear-time properties [5]. The reader may think of this product as an *attack graph* in the sense of [27].

For the rest of this section, we fix a transition system $\mathcal{S} = (S, S^0, Act, \to, \lambda)$ over Prop and we let $\mathcal{A}_\tau = (Q, C^+(\mathrm{Prop}), \delta, I, F)$.

Definition 14. *The* τ-monitoring *of $\mathcal{S}$ is the Act-labeled transition system over $\{\mathtt{win}\}$ defined by $\tau[\mathcal{S}] = (S_\tau, S_\tau^0, Act, \to_\tau, \lambda_\tau)$, where:*

- *$S_\tau = S \times Q$;*
- *S_τ^0 is the set of pairs (s_0, q) where $s_0 \in S^0$ and there exists a letter b such that $q \in \delta(I, b)$ and $\lambda(s_0) \models b$,*
- *$((s,q), a, (s',q')) \in \to_\tau$ whenever $s \overset{a}{\to} s'$ and $q' \in \delta(q, b)$ with $\lambda(s') \models b$, for some letter b,*
- $\lambda_\tau((s,q)) = \begin{cases} \{\mathtt{win}\}, & \text{if } q \in F, \\ \emptyset, & \text{otherwise.} \end{cases}$

Figure 4 illustrates the construction of $\tau^{\mathrm{e}}[\mathcal{S}^{\mathrm{e}}]$ (see also Figures 1 and 3): in the picture, we additionally recall next to each action the label read by the automaton $\mathcal{A}_{\tau^{\mathrm{e}}}$. The first label read, that is the one of initial state s_0, is written on the "start" arrow; it is $\top$ by Remark 1 since $\lambda(s_0) = \emptyset$. The label next to action a_2 is $\lambda(s_2)$, and the label next to action a_3 is $\lambda(s_3)$. The states of the automaton are dismissed.

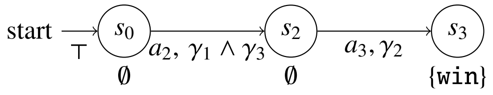

Fig. 4. The τ^{e}-monitoring of $\mathcal{S}^{\mathrm{e}}$.

The following proposition follows from Definition 14.

Proposition 3.

1) *Let $(s_0, q_1), (s_1, q_2), \dots, (s_m, q_{m+1})$ be a path in $\tau[\mathcal{S}]$. Then, the sequence of states $s_0 s_1 \dots s_m \in \Pi(\mathcal{S})$. If moreover s_0 is an initial state and $\lambda_\tau((s_m, q_{m+1})) = \{\mathtt{win}\}$, then $s_0 s_1 \dots s_m$ is a τ-attack on $\mathcal{S}$.*
2) *Reciprocally, for each τ-attack on $\mathcal{S}$ of the form $s_0 s_1 \dots s_m$, there exists a path in $\tau[\mathcal{S}]$ of the form $(s_0, q_1), (s_1, q_2), \dots, (s_m, q_{m+1})$ with $\lambda_\tau((s_m, q_{m+1})) = \{\mathtt{win}\}$.*

Proof. 1. Since $(s_0, q_1), (s_1, q_2), \dots, (s_m, q_{m+1})$ is a path in $\tau[\mathcal{S}]$, it follows from Definition 14 that $\pi = s_0 s_1 \dots s_m$ is a path in $\mathcal{S}$. In addition, for every $0 \le i \le m$, there exists $b_i \in C^+(\mathrm{Prop})$, such that $q_{i+1} \in \delta(q_i, b_i)$ and $\lambda(s_i) \models b_i$. If $\lambda_\tau((s_m, q_{m+1})) = \{\mathtt{win}\}$, then by definition $q_{m+1} \in F$, which implies that the word $u = b_0 b_1 \dots b_m \in L(\mathcal{A}_\tau) = L(\tau)$, and thus $\pi \in L(\tau)$. Clearly if s_0 is initial, π is a τ-attack on $\mathcal{S}$ by definition of a τ-attack on $\mathcal{S}$.
2. Reciprocally, let $\pi = s_0 s_1 \dots s_m \in \Pi(\mathcal{S})$. By definition of a path in $\mathcal{S}$, for every $0 \le i \le m-1$, there exists $a_i \in Act$,

such that $s_i \xrightarrow{a_i} s_{i+1}$. Since π is a τ-attack on $\mathcal{S}$, trace(π) can be embedded in some word $u = b_0 b_1 \dots b_m \in L(\tau)$, with $\lambda(s_i) \models b_i$, for every $0 \leq i \leq m$. Since $L(\tau) = L(\mathcal{A}_\tau)$, there exists a sequence of states $q_0, q_1, \dots q_{m+1}$ in $\mathcal{A}_\tau$, such that $q_{i+1} \in \delta(q_i, b_i)$ and q_{m+1} is final in $\mathcal{A}_\tau$. By Definition 14, the sequence $(s_0, q_1), (s_1, q_2), \dots, (s_m, q_{m+1})$ is a path in $\tau[\mathcal{S}]$, and since q_{m+1} is final, $\lambda_\tau((s_m, q_{m+1})) = \{\texttt{win}\}$. □

We can generalize the product to a system $(\mathcal{S}, \mathcal{Q})$ and an automaton $\mathcal{A}$: it amounts to compute the product $\mathcal{S} \times \mathcal{A}$, while adapting the quantitative features $\mathcal{Q}$ in a straightforward manner. An instance of such a product is mentioned in [8] for the product of a timed system and a Büchi automaton, and this construction underlies some functionalities of the model checker Divine [6].

We can take advantage of the monitored system to perform quantitative analysis of an attack tree at any stage of its design.

C. Early-stage quantitative analysis

Quantitative analysis approaches in attack tree-based security modeling and analysis have been widely investigated in the literature [21], [19], [11]. In all approaches, the tree is considered on its own, bereft of the system model. In this context, quantitative features are only attached to the leaves of the tree, which in all approaches are meant to represent basic action steps. The simplest way to achieve this has been studied in [21], where quantities (like duration, cost, etc.) are assigned to the leaves and, by a bottom-up computation, yield a value to the entire tree.

More recent works have considered more expressive objects reflecting quantitative features than mere values: in [19], [11], basic action steps are equipped with an operational model, such as a timed automaton or even a priced timed automaton. Again, a bottom-up computation returns a global operational model for the tree, which allows for quantitative analysis taking advantage of well-known verification methods, such as optimal reachability, model checking with temporal logic, etc.

However, despite the progress made since the pioneer approaches, such quantitative assignments to basic actions do not allow to differentiate between the values of similar actions performed in different contexts. As a consequence, the quantitative evaluation may provide an improper value to the tree.

The setting we propose offers a robustness that previous methods cannot guarantee: first, we can exploit our monitoring technique to incorporate the system into the quantitative evaluation process, and second, the evaluation can be carried out at an early-stage of the attack tree design: it is sufficient to know the reachability goal of each leaf, without the need to develop the tree until atomic actions arise. This is what we detail now.

Assume we have an attack tree τ and a system such as a PTA, say $\mathcal{T}$[5], and let us denote by $\tau[\mathcal{T}]$ the PTA resulting from the monitoring of $\mathcal{T}$ by $\mathcal{A}_\tau$ (as explained in the previous section). We can now exploit the proposition `win` in the PTA $\tau[\mathcal{T}]$ to address e.g. the cost-optimal reachability (resp. the minimum-time reachability) problem of some state labeled by `win`. According to [7], [2] (resp. [25]) this problem is decidable and an optimal path can even be synthesized.

By Proposition 3 such a path reflects some τ-attack on the system $\mathcal{T}$. The (optimal) value of this path is the right candidate for the value of the tree τ (w.r.t. the system $\mathcal{T}$), even though basic actions do not appear in the tree (they are nevertheless conveyed by the system).

Given a synthesized optimal path π^* of the system, we let trace(π^*) $\in L(\tau)$ be a *witness* of τ. In what follows, we take advantage of our ability to exhibit a witness $w \in L(\tau)$ at an early-stage of the design to guide the expert in the refinement process of some leaves.

[5]We follow the notation of [7].

VI. Guided design of attack trees

Having in hand a witness $w \in L(\tau)$ arising from the quantitative analysis described in Section V, we can determine which part of the tree, namely which set of positions, is involved when considering this very trace. Such positions are called *useful* and their charaterization is established in Theorem 3. of Subsection VI-C. This theorem relies on a propositional formula whose backbone is the so-called *membership-test* formula ϕ^τ_u of Subsection VI-B: the semantics of ϕ^τ_u, the set of models, describes all the different ways of justifying that $u \in L(\tau)$, as stated by Theorem 2 of Subsection VI-B; in particular if $u \notin L(\tau)$, formula ϕ^τ_u has no model.

The precious information on which position is useful can then be exploited to guide the designer towards useful leaves that deserve being developed further. However, as we will see in Subsection VI-D), developing a leaf, or equivalently *refining* it, may have a significant impact on the usefulness status of all leaves in the tree.

Before getting to the main Subsections VI-B, VI-C and VI-D, we make a brief recall on propositional logic.

In the rest of this section we fix an attack tree τ over Prop. W.l.o.g. we may assume that Prop $= \{\gamma \,|\, \gamma$ is a leaf of $\tau\}$.

A. Brief recall on propositional logic

In the following, let Var be a set of propositional variables, or simply *variables*.

Definition 15. *A* propositional formula *over Var is an expression conform to the following grammar: $\phi, \phi' ::= x \mid \neg\phi \mid \phi \vee \phi' \mid \phi \wedge \phi'$, where $x \in Var$. Additionally, we use standard notation $\phi \Rightarrow \phi'$ and $\phi \Leftrightarrow \phi'$ to denote $\neg\phi \vee \phi'$ and $(\phi \wedge \phi') \vee (\neg\phi \wedge \neg\phi')$, respectively. We denote by $var(\phi)$ the set of variables occurring in ϕ.*

Definition 16. *A* partial valuation *ν over Var is a partial function from Var to $\{\texttt{tt}, \texttt{ff}\}$, and we let $dom(\nu)$ be its domain. We may write $x \in \nu$ for $x \in Var$, whenever $x \in dom(\nu)$ and $\nu(x) = \texttt{tt}$.*

Two valuations ν_1 and ν_2 are compatible *if for every $x \in dom(\nu_1) \cap dom(\nu_2)$, $\nu_1(x) = \nu_2(x)$. Given two compatible valuations ν_1 and ν_2, we write $\nu_1 \cup \nu_2$ for the valuation whose* *domain is $dom(\nu_1) \cup dom(\nu_2)$, defined by: for all $x \in dom(\nu_1)$, $(\nu_1 \cup \nu_2)(x) = \nu_1(x)$ and for all $x \in dom(\nu_2)$, $(\nu_1 \cup \nu_2)(x) = \nu_2(x)$.*

Definition 17. *Let ϕ be a propositional formula over Var and ν be a partial valuation over Var. The valuation ν is a* model *of ϕ, written $\nu \models \phi$, whenever the following holds (by induction over ϕ):*

- *$\nu \models x \in Var$ if, and only if $x \in \nu$,*
- *$\nu \models \neg\phi$ if, and only if $\nu \not\models \phi$,*
- *$\nu \models \phi \wedge \phi'$ if, and only if $\nu \models \phi$ and $\nu \models \phi'$,*
- *$\nu \models \phi \vee \phi'$ if, and only if $\nu \models \phi$ or $\nu \models \phi'$.*

Lemma 1. *Let ϕ_1, ϕ_2 be two propositional formulas over Var, and let ν_1, ν_2 be two compatible valuations with $\nu_1 \models \phi_1$ and $\nu_2 \models \phi_2$. If $var(\phi_2) \cap dom(\nu_1) = \emptyset$ and $var(\phi_1) \cap dom(\nu_2) = \emptyset$, then $\nu_1 \cup \nu_2 \models \phi_1 \wedge \phi_2$.*

The proof of the above lemma is trivial and thus omitted. For the rest of the section, we let

$$Var := \{x^p_{(i,j)} \mid p \in \{1,2\}^* \text{ and } i \leq j \in \mathbb{N}\}[6].$$

Intuitively, for each propositional variable $x^p_{(i,j)}$, p will range over the set of positions of the attack tree τ, while i and j will serve to denote a subword $u[i,j]$ of some trace u of the system. The variable $x^p_{(i,j)}$, when set to true, reflects the fact $u[i,j] \in L(\tau)$.

As we will reason by induction over the subtrees of τ, we need some simple mechanism to rename the variables so that they refer to the correct position in the global context τ. This is achieved by the notion of *p-shift*: given a fornula ϕ over Var and $p \in \{1,2\}^*$, the *p-shift* of ϕ is the formula $\phi{\downarrow_p}$ obtained by replacing each variable $x^q_{(i,j)}$ in ϕ by the variable $x^{p.q}_{(i,j)}$.

B. Membership-test formula

Given a word u over $C^+(\text{Prop})$, we define a propositional formula ϕ^τ_u over Var, that we call the *membership-test* of u in τ, that holds whenever $u \in L(\tau)$ (Theorem 2).

Formula ϕ^τ_u expresses that $u \in L(\tau)$, hence it requires variable $x^\epsilon_{(1,n)}$ to be true, but it also has a second conjunct, called the *consistency-test* formula, and written ψ^τ_u, that axiomatizes the semantics of the whole tree: first, it needs to reflect the semantics of all operators in internal positions of the tree nodes, which is achieved by the so-called *junction* formulas of the form $\Gamma^{\mathtt{OP},p}_{i,j}$ (see Definition 18). Second, it needs to reflect the semantics of each leaf of the tree: namely that for a leaf γ, variable $x^p_{(i,j)}$ must be false if $u[i,j] \notin L(\gamma)$, otherwise said if $u(j) \not\models \gamma$.

We first introduce junction formulas, for each kind of operator `OP`.

Definition 18. *Let u be a word over $C^+(\text{Prop})$ of size n. For each $p \in Pos_{\mathtt{OP}}(\tau) \setminus \ell(\tau)$ and $1 \leq i \leq j \leq n$, we let $\Gamma^{OP,p}_{i,j}$, called the* junction formula *at position p between i and j, be defined by:*

- $\Gamma^{OR,p}_{i,j} := \left(x^p_{(i,j)} \Leftrightarrow x^{p.1}_{(i,j)} \vee x^{p.2}_{(i,j)}\right)$
- $\Gamma^{SAND,p}_{i,j} := \left(x^p_{(i,j)} \Leftrightarrow \bigvee_{i \leq k \leq j} x^{p.1}_{(i,k)} \wedge \left(x^{p.2}_{(k,j)} \vee x^{p.2}_{(k+1,j)}\right)\right)$
- $\Gamma^{AND,p}_{i,j} := \left(x^p_{(i,j)} \Leftrightarrow \bigvee_{i \leq k,l \leq j, k \leq l+1} \left(x^{p.1}_{(i,l)} \wedge x^{p.2}_{(k,j)}\right) \vee \left(x^{p.2}_{(i,l)} \wedge x^{p.1}_{(k,j)}\right)\right)$.

[6]For technical reasons, in formulas we may have expressions of the form $x^p_{(i+1,i)}$, that are not in Var, and that are always interpreted as *false*.

We can now define the *consistency-test* formula ψ^τ_u.

Definition 19. *Let u be a word over $C^+(\text{Prop})$ of size n. The* consistency-test *formula for u in τ, is the formula ψ^τ_u defined by (induction on τ):*

- $\psi^\gamma_u := \bigwedge_{1 \leq i \leq j \leq n, u(j) \not\models \gamma} \neg x^\epsilon_{(i,j)}$
- $\psi^{OP(\tau_1,\tau_2)}_u := \psi^{\tau_1}_u{\downarrow_1} \wedge \psi^{\tau_2}_u{\downarrow_2} \wedge \bigwedge_{1 \leq i \leq j \leq n} \Gamma^{\mathtt{OP},\epsilon}_{i,j}$.

Finally, we obtain the *membership-test* formula ϕ^τ_u as announced.

Definition 20. *The* membership-test formula *for u in τ is defined by $\phi^\tau_u := x^\epsilon_{(1,n)} \wedge \psi^\tau_u$.*

Example 4 illustrates the membership-test formula.

Example 4. *Let τ^e be the attack tree from Figure 1. Let u^e be the word $\top(\gamma_1 \wedge \gamma_3)\gamma_2$ of size 3. We saw in Example 3 that $u^e \in L(\tau^e)$.*

By Definition 20, $\phi^{\tau^e}_u = x^\epsilon_{(1,3)} \wedge \psi^\tau_u$, and by Definition 19,

$$\psi^{AND(\tau^e|_1,\tau^e|_2)}_{u^e} = \psi^{SAND(\gamma_1,\gamma_2)}_{u^e}{\downarrow_1} \wedge \psi^{OR(\gamma_3,\gamma_4)}_{u^e}{\downarrow_2} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{AND,\epsilon}_{i,j}$$

$$\textit{with} \qquad \psi^{SAND(\gamma_1,\gamma_2)}_{u^e} = \psi^{\gamma_1}_{u^e}{\downarrow_1} \wedge \psi^{\gamma_2}_{u^e}{\downarrow_2} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{SAND,\epsilon}_{i,j}$$

$$\textit{and} \qquad \psi^{OR(\gamma_3,\gamma_4)}_{u^e} = \psi^{\gamma_3}_{u^e}{\downarrow_1} \wedge \psi^{\gamma_4}_{u^e}{\downarrow_2} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{OR,\epsilon}_{i,j}$$

so by applying the shifting, we have $\psi^{AND(\tau^e|_1,\tau^e|_2)}_{u^e} = \psi^{\gamma_1}_{u^e}{\downarrow_{11}} \wedge \psi^{\gamma_2}_{u^e}{\downarrow_{12}} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{SAND,1}_{i,j} \wedge \psi^{\gamma_3}_{u^e}{\downarrow_{21}} \wedge \psi^{\gamma_4}_{u^e}{\downarrow_{22}} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{OR,2}_{i,j} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{AND,\epsilon}_{i,j}$.

For all $i \in \{1,2,3,4\}$, we have $\psi^{\gamma_i}_{u^e} = \bigwedge_{1 \leq i \leq j \leq 3, u^e(j) \not\models \gamma_i} \neg x^\epsilon_{(i,j)}$ which, after being applied to u^e and shifted to the position of the corresponding leaf, gives

- $\psi^{\gamma_1}_{u^e}{\downarrow_{11}} = \neg x^{11}_{(1,3)} \wedge \neg x^{11}_{(2,3)} \wedge \neg x^{11}_{(3,3)}$
- $\psi^{\gamma_2}_{u^e}{\downarrow_{12}} = \neg x^{12}_{(1,2)} \wedge \neg x^{12}_{(2,2)}$
- $\psi^{\gamma_3}_{u^e}{\downarrow_{21}} = \neg x^{21}_{(1,3)} \wedge \neg x^{21}_{(2,3)} \wedge \neg x^{21}_{(3,3)}$
- $\psi^{\gamma_4}_{u^e}{\downarrow_{22}} = \neg x^{22}_{(1,2)} \wedge \neg x^{22}_{(2,2)} \wedge \neg x^{22}_{(1,3)} \wedge \neg x^{22}_{(2,3)} \wedge \neg x^{22}_{(3,3)}$.

So, $\phi^{\tau^e}_{u^e} = x^\epsilon_{(1,3)} \wedge \neg x^{11}_{(1,3)} \wedge \neg x^{11}_{(2,3)} \wedge \neg x^{11}_{(3,3)} \wedge \neg x^{12}_{(1,2)} \wedge \neg x^{12}_{(2,2)} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{SAND,1}_{i,j} \wedge \neg x^{21}_{(1,3)} \wedge \neg x^{21}_{(2,3)} \wedge \neg x^{21}_{(3,3)} \wedge \neg x^{22}_{(1,2)} \wedge \neg x^{22}_{(2,2)} \wedge \neg x^{22}_{(1,3)} \wedge \neg x^{22}_{(2,3)} \wedge \neg x^{22}_{(3,3)} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{OR,2}_{i,j} \wedge \bigwedge_{1 \leq i \leq j \leq 3} \Gamma^{AND,\epsilon}_{i,j}$

Now, we can expand the definition of $\Gamma^{AND,\epsilon}_{i,j}$, $\Gamma^{SAND,1}_{i,j}$, and

$\Gamma^{OR,2}_{i,j}$, *which gives*

$$\begin{aligned}\phi^{\tau^e}_{u^e} = & x^{\epsilon}_{(1,3)} \wedge \neg x^{11}_{(1,3)} \wedge \neg x^{11}_{(2,3)} \wedge \neg x^{11}_{(3,3)} \\ & \wedge \neg x^{12}_{(1,2)} \wedge \neg x^{12}_{(2,2)} \\ & \wedge \bigwedge_{1\le i\le j\le 3} \Big(x^1_{(i,j)} \Leftrightarrow \bigvee_{i\le k\le j} x^{11}_{(i,k)} \wedge \big(x^{22}_{(k,j)} \vee x^{22}_{(k+1,j)}\big)\Big) \\ & \wedge \neg x^{21}_{(1,3)} \wedge \neg x^{21}_{(2,3)} \wedge \neg x^{21}_{(3,3)} \\ & \wedge \neg x^{22}_{(1,2)} \wedge \neg x^{22}_{(2,2)} \wedge \neg x^{22}_{(1,3)} \wedge \neg x^{22}_{(2,3)} \wedge \neg x^{22}_{(3,3)} \\ & \wedge \bigwedge_{1\le i\le j\le 3} \big(x^2_{(i,j)} \Leftrightarrow x^{21}_{(i,j)} \vee x^{22}_{(i,j)}\big) \\ & \wedge \bigwedge_{1\le i\le j\le 3} \Big(x^{\epsilon}_{(i,j)} \Leftrightarrow \bigvee_{i\le k,l\le j,k\le l+1} \big(x^1_{(i,l)} \wedge x^2_{(k,j)}\big) \vee \big(x^2_{(i,l)} \wedge x^1_{(k,j)}\big)\Big)\end{aligned}$$

One can verify that the valuation assigning tt *to variables* $x^{1.1}_{(1,2)}, x^{1.2}_{(2,3)}, x^{2.1}_{(1,2)}, x^1_{(1,3)}, x^2_{(1,2)}, x^{\epsilon}_{(1,3)}$ *and* ff *to the other ones is a model of* $\phi^{\tau^e}_{u^e}$*. Recall that* $u \in L(\tau^e)$ *(see Example 3). On the contrary, for the word* $v^e = \top\gamma_4\gamma_2$*, notice that variables* $x^{1.1}_{(i,k)}$ *must be assigned* ff *for every* $1 \le i \le k \le 3$*, since* γ_1 *never occurs in* v*. Therefore, because* $x^1_{(i,j)} \Leftrightarrow \bigvee_{i\le k\le j} x^{1.1}_{(i,k)} \wedge \big(x^{1.2}_{(k,j)} \vee x^{1.2}_{(k+1,j)}\big)$*, variables* $x^1_{(i,j)}$ *must also be assigned* ff*, for all* $1 \le i \le j \le 3$*. Now, since* $x^{\epsilon}_{(i,j)} \Leftrightarrow \bigvee_{i\le k,l\le j,k\le l+1} \big(x^1_{(i,l)} \wedge x^2_{(k,j)}\big) \vee \big(x^2_{(i,l)} \wedge x^1_{(k,j)}\big)$*,* $x^{\epsilon}_{(i,\ell)}$ *must also be assigned* ff*, which is not compatible with the first conjunct* $x^{\epsilon}_{(1,3)}$ *of the membership formula* $\phi^{\tau^e}_v$*. As a consequence, formula* $\phi^{\tau^e}_v$ *has no model. Recall that* $v \notin L(\tau^e)$ *(see Example 3).*

All the facts illustrated in Example 4 are correlated as stated in Theorem 2 which relates the membership-test formula satisfiability and the membership of words in the trace semantics of an attack tree. This theorem is central for the approach developed in our work. It has two main repercussions: 1) it gives a SAT-based algorithm to verify that a trace is in the language of the tree (an alternative procedure for the membership problem than the one based on the trace automaton $\mathcal{A}_\tau$), and 2) it is a milestone for the proof of Theorem 3 which gives us the criterion to identify useful positions.

Theorem 2. *Let u be a word over* $C^+(\mathrm{Prop})$ *of size n. The membership-test formula for u in* τ *has a model if, and only if,* $u \in L(\tau)$*.*

The rest of this section is dedicated to proving Theorem 2. We will actually show a more general equivalence stated in Equation (1)

$$\phi^{\tau,p}_{u[i,j]} \text{ has a model if, and only if } u[i,j] \in L(\tau_{|p}), \tag{1}$$

where $\phi^{\tau,p}_{u[i,j]} := x^p_{(i,j)} \wedge \psi^{\tau,p}_{u[i,j]}$, with

- $\psi^{\gamma,p}_{u[i,j]} := \bigwedge_{i\le i'\le j'\le j, u(j')\not\models\gamma} \neg x^p_{(i',j')}$
- if $\tau(p) = \mathtt{OP}$, then $\psi^{\tau,p}_{u[i,j]} := \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]} \wedge \bigwedge_{i\le i'\le j'\le j} \Gamma^{\mathtt{OP},p}_{i',j'}$.

In particular, it is easy to verify that:

$$\psi^{\tau,\epsilon}_{u[1,n]} = \psi^{\tau}_u \text{ (and therefore } \phi^{\tau,\epsilon}_{u[1,n]} = \phi^{\tau}_u). \tag{2}$$

The particular case of Equations (1) and (2), where $p = \epsilon$, $i = 1$ and $j = n$ entails the statement of Theorem 2.

In order to show Equation (1), we establish four helpful lemmas. Lemma 2 shows that the recursive definition of $\psi^{\tau,p}_{u[i,j]}$ can be flattened, while Lemma 3 establishes a relationship between such formulas. Lemmas 4 and 5 state conditions for the boundaries i and j to be safely changed.

Lemma 2. *The consistency-test formula* $\psi^{\tau,p}_{u[i,j]}$ *is equivalent to*

$$\bigwedge_{\substack{q\in Pos(\tau)\setminus\ell(\tau)\\ p\le q\\ i\le i'\le j'\le j}} \Gamma^{\tau(q),q}_{i',j'} \wedge \bigwedge_{\substack{q\in\ell(\tau)\\ p\le q\\ i\le i'\le j'\le j\\ u(j')\not\models\tau(q)}} \neg x^q_{(i',j')}. \tag{3}$$

Proof. We reason by induction over the height[7] of $\tau_{|p}$.

If $\tau_{|p}$ is a leaf, say $\tau(p) = \gamma$, then it is clear by definitions of $\psi^{\gamma,p}_{u[i,j]}$ and by the fact that the first conjunct of Equation (3) vanishes.

Otherwise, let $\tau(p) = \mathtt{OP}$. By definition, $\psi^{\tau,p}_{u[i,j]} = \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]} \wedge \bigwedge_{i\le i'\le j'\le j} \Gamma^{\mathtt{OP},p}_{i',j'}$, and by induction hypothesis,

$$\psi^{\tau,p.1}_{u[i,j]} = \bigwedge_{\substack{q\in\mathrm{Pos}(\tau)\setminus\ell(\tau)\\ p.1\le q\\ i\le i'\le j'\le j}} \Gamma^{\tau(q),q}_{i',j'} \wedge \bigwedge_{\substack{q\in\ell(\tau)\\ p.1\le q\\ i\le i'\le j'\le j\\ u(j')\not\models\tau(q)}} \neg x^q_{(i',j')}$$

$$\text{and} \quad \psi^{\tau,p.2}_{u[i,j]} = \bigwedge_{\substack{q\in\mathrm{Pos}(\tau)\setminus\ell(\tau)\\ p.2\le q\\ i\le i'\le j'\le j}} \Gamma^{\tau(q),q}_{i',j'} \wedge \bigwedge_{\substack{q\in\ell(\tau)\\ p.2\le q\\ i\le i'\le j'\le j\\ u(j')\not\models\tau(q)}} \neg x^q_{(i',j')}.$$

Now, gathering all positions of the formulas and noticing that $\epsilon \in \mathrm{Pos}(\tau) \setminus \ell(\tau)$, we can rewrite $\psi^{\tau,p}_{u[i,j]}$ as the formula of Equation (3).

□

Relying on Lemma 2, it easy to show the following technical lemma (whose proof is omitted).

Lemma 3. *Let* $i \le i' \le j' \le j$*. The formula* $\psi^{\tau,p}_{u[i,j]}$ *is equivalent to*

$$\psi^{\tau,p}_{u[i',j']} \wedge \Big(\bigwedge_{\substack{[i'',j'']\subseteq[i,j]\\ [i'',j'']\not\subseteq[i',j']\\ p\le q\notin\ell(\tau)}} \Gamma^{\tau(q),q}_{i'',j''} \wedge \bigwedge_{\substack{[i'',j'']\subseteq[i,j]\\ [i'',j'']\not\subseteq[i',j']\\ p\le q\in\ell(\tau)\\ u(j'')\not\models\tau(q)}} \neg x^q_{(i'',j'')}\Big). \tag{4}$$

Lemma 4. *Let* $i \le i' \le j' \le j$*. If* $\nu \models \psi^{\tau,p}_{u[i,j]}$*, then* $\nu \models \psi^{\tau,p}_{u[i',j']}$*.*

Proof. By Lemma 3, $\psi^{\tau,p}_{u[i',j']}$ is a conjunct of $\psi^{\tau,p}_{u[i,j]}$, which concludes. □

Lemma 5. *Let* $i \le i' \le j' \le j$*. If* $\nu \models \psi^{\tau,p}_{u[i',j']}$ *with* $dom(\nu) = var(\psi^{\tau,p}_{u[i',j']})$*, then* $\nu \models \psi^{\tau,p}_{u[i,j]}$ *(where* ν *is interpreted over* $var(\psi^{\tau,p}_{u[i,j]})$ *by assigning all extra variables to false).*

Proof. We use Lemma 3 and remark that the right-hand conjunct of Equation (4) is satisfied by the valuation $\emptyset$ (denoting the valuation with an empty domain) and involves only propositions that do not appear in $\psi^{\tau,p}_{u[i',j']}$.

[7]The height is the length of the longest branch.

By Lemma 1, $\nu \cup \emptyset = \nu \models \psi^{\tau,p}_{u[i,j]}$. ◻

We now turn to the proof of Equation (1).

$\Rightarrow$) Suppose that $\phi^{\tau,p}_{u[i,j]}$ has a model, say $\nu \models \phi^{\tau,p}_{u[i,j]}$. We reason by induction on the height of $\tau_{|p}$.

If $\tau_{|p}$ is a leaf, say $\tau(p) = \gamma$, then $\nu \models x^p_{(i,j)} \wedge \bigwedge_{i \le i' \le j' \le j, u(j') \not\models \gamma} \neg x^p_{(i',j')}$. Since $x^p_{(i,j)} \in \nu$, proposition $x^p_{(i,j)}$ is not among the propositions $x^p_{(i',j')}$ such that $u(j') \not\models \gamma$ (otherwise the formula would be contradictory), thus $u(j) \models \gamma$ which entails $u[i,j] \in L(\gamma)$.

Otherwise, $\tau_{|p} = \texttt{OP}(\tau_{|p.1}, \tau_{|p.2})$.

- if $\texttt{OP} = \texttt{OR}$, then $\nu \models x^p_{(i,j)} \wedge \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]} \wedge \bigwedge_{i \le i' \le j' \le j} \Gamma^{\texttt{OR},p}_{i',j'}$. So $x^p_{(i,j)} \in \nu$, and as $\nu \models \Gamma^{\texttt{OR},p}_{i,j}$, we have either $x^{p.1}_{(i,j)} \in \nu$ or $x^{p.2}_{(i,j)} \in \nu$. Assume that $x^{p.1}_{(i,j)} \in \nu$, as the other case is symmetric. We have $x^{p.1}_{(i,j)} \in \nu$ and $\nu \models \psi^{\tau,p.1}_{u[i,j]}$ so $\nu \models x^{p.1}_{(i,j)} \wedge \psi^{\tau,p.1}_{u[i,j]} = \phi^{\tau,p.1}_{u[i,j]}$. By induction hypothesis, $u[i,j] \in L(\tau_{|p.1})$. As $L(\tau_{|p}) = L(\tau_{|p.1}) \cup L(\tau_{|p.2})$, we have $u[i,j] \in L(\tau_{|p})$.
- if $\texttt{OP} = \texttt{SAND}$, then $\nu \models x^p_{(i,j)} \wedge \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]} \wedge \bigwedge_{i \le i' \le j' \le j} \Gamma^{\texttt{SAND},p}_{i,j}$. So $x^p_{(i,j)} \in \nu$, and as $\nu \models \Gamma^{\texttt{SAND},p}_{i,j}$, we have that for some $i \le k \le j$, $x^{p.1}_{(i,k)} \in \nu$ and either $x^{p.2}_{(k,j)} \in \nu$ or $x^{p.2}_{(k+1,j)} \in \nu$. Assume that $x^{p.2}_{(k,j)} \in \nu$, as the other case is symmetric. We have $x^{p.1}_{(i,k)} \in \nu$ and $\nu \models \psi^{\tau,p.1}_{u[i,j]}$ and by Lemma 4, $\nu \models \psi^{\tau,p.1}_{u[i,k]}$, so $\nu \models x^{p.1}_{(i,k)} \wedge \psi^{\tau,p.1}_{u[i,k]} = \phi^{\tau,p.1}_{u[i,k]}$. By induction hypothesis, $u[i,k] \in L(\tau_{|p.1})$. We have $x^{p.2}_{(k,j)} \in \nu$ and $\nu \models \psi^{\tau,p.2}_{u[i,j]}$ and by Lemma 4, $\nu \models \psi^{\tau,p.2}_{u[k,j]}$, so $\nu \models x^{p.2}_{(k,j)} \wedge \psi^{\tau,p.2}_{u[k,j]} = \phi^{\tau,p.2}_{u[k,j]}$. By induction hypothesis, $u[k,j] \in L(\tau_{|p.2})$. So $u[i,j] = u[i,k] \odot u[k,j] \in L(\tau_{|p.1}) \odot L(\tau_{|p.2}) = L(\tau_{|p})$.
- if $\texttt{OP} = \texttt{AND}$, then $\nu \models x^p_{(i,j)} \wedge \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]} \wedge \bigwedge_{i \le i' \le j' \le j} \Gamma^{\texttt{AND},p}_{i,j}$. So $x^p_{(i,j)} \in \nu$, and as $\nu \models \Gamma^{\texttt{AND},p}_{i,j}$, we have that for some $i \le k, l \le j$ with $k \le l+1$, either $x^{p.1}_{(i,l)} \in \nu$ and $x^{p.2}_{(k,j)} \in \nu$, or $x^{p.2}_{(i,l)} \in \nu$ and $x^{p.1}_{(k,j)} \in \nu$. Assume that $x^{p.1}_{(i,l)} \in \nu$ and $x^{p.2}_{(k,j)} \in \nu$, as the other case is symmetric. We have $x^{p.1}_{(i,l)} \in \nu$ and $\nu \models \psi^{\tau,p.1}_{u[i,j]}$ and by Lemma 4, $\nu \models \psi^{\tau,p.1}_{u[i,l]}$, so $\nu \models x^{p.1}_{(i,l)} \wedge \psi^{\tau,p.1}_{u[i,l]} = \phi^{\tau,p.1}_{u[i,l]}$. By induction hypothesis, $u[i,l] \in L(\tau_{|p.1})$. We have $x^{p.2}_{(k,j)} \in \nu$ and $\nu \models \psi^{\tau,p.2}_{u[i,j]}$ and by Lemma 4, $\nu \models \psi^{\tau,p.2}_{u[k,j]}$, so $\nu \models x^{p.2}_{(k,j)} \wedge \psi^{\tau,p.2}_{u[k,j]} = \phi^{\tau,p.2}_{u[k,j]}$. By induction hypothesis, $u[k,j] \in L(\tau_{|p.2})$. So $u[i,j] = u[i,k] \otimes u[k,j] \in L(\tau_{|p.1}) \otimes L(\tau_{|p.2}) = L(\tau_{|p})$.

$\Leftarrow$): suppose that $u[i,j] \in L(\tau_{|p})$. We reason by induction on the height of $\tau_{|p}$.

If $\tau_{|p}$ is a leaf, say $\tau(p) = \gamma$, then $\{x^p_{(i,j)}\} \models \phi^{\tau,p}_{u[i,j]}$ because $u[i,j] \in L(\tau_{|p})$ implies that $u(j) \models \gamma$, so $\neg x^p_{(i,j)}$ does not appear in $\psi^{\tau,p}_{u[i,j]}$.

Otherwise, $\tau_{|p} = \texttt{OP}(\tau_{|p.1}, \tau_{|p.2})$.

- if $\texttt{OP} = \texttt{OR}$, then $u[i,j] \in L(\tau_{|p.1}) \cup L(\tau_{|p.2})$ so w.l.o.g, assume that $u[i,j] \in L(\tau_{|p.1})$ (the case $u[i,j] \in L(\tau_{|p.2})$ is similar). By induction hypothesis, the formula $\phi^{\tau,p.1}_{u[i,j]}$ is satisfiable, so let $\nu' \models \phi^{\tau,p.1}_{u[i,j]}$ with $dom(\nu') = var(\psi^{\tau,p.1}_{u[i,j]})$. We have immediately that $\nu' \models \psi^{\tau,p.1}_{u[i,j]}$, and by Lemma 1 and because $var(\psi^{\tau,p.1}_{u[i,j]}) \cap var(\psi^{\tau,p.2}_{u[i,j]}) = \emptyset$, and $\emptyset \models \psi^{\tau,p.2}_{u[i,j]}$, we have $\nu' \models \psi^{\tau,p.2}_{u[i,j]}$. Let $\nu = \nu' \cup \{x^p_{(i',j')} \mid x^{p.1}_{(i',j')} \in \nu', i \le i' \le j' \le j\}$. Because $\nu' \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$, and $x^{p.1}_{(i,j)} \in \nu'$, we have $\nu \models x^p_{(i,j)} \wedge \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$. We now show that $\nu \models \Gamma^{\texttt{OR},p}_{i',j'}$ for all $i \le i' \le j' \le j$. Let $i \le i' \le j' \le j$. We have $x^p_{(i',j')} \in \nu$ if, and only if $x^{p.1}_{(i',j')} \in \nu$ by definition of ν. Additionally, recall that $var(\psi^{\tau,p.2}_{u[i,j]}) \cap dom(\nu') = \emptyset$, so as we have $x^{p.2}_{(i,j)} \in var(\psi^{\tau,p.2}_{u[i,j]})$ clearly $x^{p.2}_{(i,j)} \notin \nu'$ and then $x^{p.2}_{(i,j)} \notin \nu$, which entails $\nu \models \Gamma^{\texttt{OR},p}_{i',j'}$
- if $\texttt{OP} = \texttt{SAND}$, then $u[i,j] \in L(\tau_{|p.1}) \odot L(\tau_{|p.2})$ so w.l.o.g, assume that for some k, we have $u[i,k] \in L(\tau_{|p.1})$ and $u[k,j] \in L(\tau_{|p.2})$ (the case $u[k+1,j] \in L(\tau_{|p.2})$ is similar). By induction hypothesis, the formulas $\phi^{\tau,p.1}_{u[i,k]}$ and $\phi^{\tau,p.2}_{u[k,j]}$ are satisfiable, so let $\nu_1 \models \phi^{\tau,p.1}_{u[i,k]}$ and $\nu_2 \models \phi^{\tau,p.2}_{u[k,j]}$ with $dom(\nu_1) = var(\psi^{\tau,p.1}_{u[i,j]})$ and $dom(\nu_2) = var(\psi^{\tau,p.2}_{u[i,j]})$. By Lemma 5 we have $\nu_1 \models \psi^{\tau,p.1}_{u[i,j]}$ and $\nu_2 \models \psi^{\tau,p.2}_{u[i,j]}$, and because $var(\psi^{\tau,p.1}_{u[i,j]}) \cap var(\psi^{\tau,p.2}_{u[i,j]}) = \emptyset$, we have by Lemma 1 that $\nu_1 \cup \nu_2 \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$. Let $\nu = \nu_1 \cup \nu_2 \cup \{x^p_{(i',j')} \mid \text{ for some } k, x^{p.1}_{(i',k)} \in \nu_1 \text{ and } x^{p.2}_{(k,j')} \in \nu_2 \text{ or } x^{p.2}_{(k+1,j')} \in \nu_2\}$. Clearly, $var(\psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]})$ contains no variable of the form $x^p_{(i',j')}$, so $\nu \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$. We now show that for all $i \le i' \le j' \le j$, we have $\nu \models \Gamma^{\texttt{SAND},p}_{i,j}$. Let $i \le i' \le j' \le j$. We have $x^p_{(i',j')} \in \nu$ if, and only if for some k, $x^{p.1}_{(i',k)} \in \nu_1$ and $x^{p.2}_{(k,j')} \in \nu_2$ or $x^{p.2}_{(k+1,j')} \in \nu_2$ by definition of ν, so $\nu \models x^p_{(i',j')} \Leftrightarrow \bigvee_{i' \le k \le j'} x^{p.1}_{(i',k)} \wedge (x^{p.2}_{(k,j')} \vee x^{p.2}_{(k+1,j')})$ so $\nu \models \Gamma^{\texttt{SAND},p}_{i',j'}$. Additionally, we have that for some $k \in [i,j]$, $x^{p.1}_{(i,k)} \in \nu_1$ and $x^{p.2}_{(k,j)} \in \nu_2$, so by definition of ν we have $x^p_{(i,j)} \in \nu$. As we also have $\nu \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$ and $\nu \models \bigwedge_{i \le i' \le j' \le j} \Gamma^{\texttt{SAND},p}_{i',j'}$, we have $\nu \models \phi^{\tau,p}_{u[i,j]}$.
- if $\texttt{OP} = \texttt{AND}$, then $u[i,j] \in L(\tau_{|p.1}) \otimes L(\tau_{|p.2})$ so w.l.o.g, assume that for some k, l we have $u[i,l] \in L(\tau_{|p.1})$ and $u[k,j] \in L(\tau_{|p.2})$ (the case $u[i,l] \in L(\tau_{|p.2})$ and $u[k,j] \in L(\tau_{|p.1})$ is similar). By induction hypothesis, the formulas $\phi^{\tau,p.1}_{u[i,l]}$ and $\phi^{\tau,p.2}_{u[k,j]}$ are satisfiable, so let $\nu_1 \models \phi^{\tau,p.1}_{u[i,l]}$ and $\nu_2 \models \phi^{\tau,p.2}_{u[k,j]}$ with $dom(\nu_1) = var(\psi^{\tau,p.1}_{u[i,j]})$ and $dom(\nu_2) = var(\psi^{\tau,p.2}_{u[i,j]})$. By Lemma 5, we have $\nu_1 \models \psi^{\tau,p.1}_{u[i,j]}$ and $\nu_2 \models \psi^{\tau,p.2}_{u[i,j]}$, and because $var(\psi^{\tau,p.1}_{u[i,j]}) \cap var(\psi^{\tau,p.2}_{u[i,j]}) = \emptyset$, we have by Lemma 1 that $\nu_1 \cup \nu_2 \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$. Let $\nu = \nu_1 \cup \nu_2 \cup \{x^p_{(i',j')} \mid \text{ for some } k, l \in [i,j] \text{ with } k \le l+1, x^{p.1}_{(i',l)} \in \nu_1 \text{ and } x^{p.2}_{(k,j')} \in \nu_2, \text{ or } x^{p.2}_{(i',l)} \in \nu_1 \text{ and } x^{p.1}_{(k,j')} \in \nu_2\}$. Clearly, $var(\psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]})$ contains no variable of the form $x^p_{(i',j')}$, so we have $\nu \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$. We now show that for all $i \le i' \le j' \le j$, we have $\nu \models \Gamma^{\texttt{AND},p}_{i',j'}$. Let $i \le i' \le j' \le j$. We have $x^p_{(i',j')} \in \nu$ if, and only if for some $k, l \in [i,j]$ with $k \le l+1$, $x^{p.1}_{(i',l)} \in \nu_1$ and $x^{p.2}_{(k,j')} \in \nu_2$, or $x^{p.2}_{(i',l)} \in \nu_1$ and $x^{p.1}_{(k,j')} \in \nu_2$ by definition of ν, so $\nu \models x^p_{(i',j')} \Leftrightarrow (x^{p.1}_{(i',l)} \wedge x^{p.2}_{(k,j')}) \vee (x^{p.2}_{(i',l)} \wedge x^{p.1}_{(k,j')})$, so $\nu \models \Gamma^{\texttt{AND},p}_{i',j'}$. Additionally, we have that for some $k, l \in [i,j]$ with $k \le l+1$, we have $x^{p.1}_{(i,l)} \in \nu_1$ and $x^{p.2}_{(k,j)} \in \nu_2$, so the

definition of ν entails that $x^{p}_{(i,j)} \in \nu$. As we also have $\nu \models \psi^{\tau,p.1}_{u[i,j]} \wedge \psi^{\tau,p.2}_{u[i,j]}$ and $\nu \models \bigwedge_{i \leq i' \leq j' \leq j} \Gamma^{\mathtt{AND},p}_{i',j'}$, we have that $\nu \models \phi^{\tau,p}_{u[i,j]}$.

Now that the membership-test formula plays its role, we can consider words that belong to $L(\tau)$, that is τ-attacks on $\mathcal{S}$, like witnesses. Such words have a satisfiable membership-test formula (by Theorem 2) that we over constrain to test whether a given position in the tree plays a role in this membership.

C. Useful positions

In order to motivate Definition 21 of useful position, assume τ is of the form $\mathtt{OR}(\tau_1, \tau_2)$ and take $u \in L(\tau)$. By the semantics of $\mathtt{OR}$ we know that either $u \in L(\tau_1)$ or $u \in L(\tau_2)$. For example, in case when $u \in L(\tau_1)$ but $u \notin L(\tau_2)$, it is natural to think of the subtree τ_2 as useless for u, or equivalently of position 2 as useless for u, and to think of position 1 as useful for u. Regarding the membership of word u, considering the tree τ or τ_1, does not make any difference. The usefulness of position 1 for $u \in L(\tau)$ can be rephrased as the usefulness of position ϵ for $u \in L(\tau[\epsilon \leftarrow \tau_1])$, since it is equal to τ_1.

Also, because 2 is useless for u, so are all positions below 2 in τ. Therefore, for a position to be useful, it is necessary that this is also the case for all its ancestor positions.

Our illustration for the notion of usefulness is a toy example where the parent of the useful position is the root position. This principle should be extended to all arbitrary positions p whose parent is an $\mathtt{OR}$-position: this is captured by the first condition of Definition 21 that $u \in L(\tau[par(p) \leftarrow \tau_{|p}])$ and $par(p)$ is useful for u in $\tau[par(p) \leftarrow \tau_{|p}]$.

The second condition of Definition 21 relates to positions whose parent q is not an $\mathtt{OR}$-position: since for position q both $q.1$ and $q.2$ contribute to the membership of a word in $L(\tau_{|q})$, it is natural to let $q.1$ and $q.2$ be useful (provided position q itself is useful).

Definition 21 (Useful positions)**.** *Let τ be an attack tree over* Prop*, and let $u \in L(\tau)$. A position $p \in Pos(\tau)$ is useful for u in τ if either $p = \epsilon$, or $p \neq \epsilon$ and $par(p)$ is useful for u in τ and one of the two following conditions holds:*

- *$\tau(par(p)) = \mathtt{OR}$ and $u \in L(\tau[par(p) \leftarrow \tau_{|p}])$ and $par(p)$ is useful for u in $\tau[par(p) \leftarrow \tau_{|p}]$, or*
- *$\tau(par(p)) \neq \mathtt{OR}$.*

A position is useless *if it is not useful.*

We now give a characterization of useful positions for u in τ by the satisfiability of the *usefulness-test* formula of p for u in τ.

Theorem 3. *Let $u \in L(\tau)$ of size n and $p \in Pos(\tau)$. Then, p is useful for u in τ if, and only if $\Upsilon^{\tau,p}_{u} := \phi^{\tau}_{u} \wedge \bigwedge_{q \leq p} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)}$, called the* usefulness-test *formula of p for u in τ, has a model.*

The rest of this section is the proof of Theorem 3.

We first state Lemma 6 (whose proof can be found in Appendix B) that exhibits a relationship between the membership-test formulas of τ and of a tree obtained by substituting in τ an $\mathtt{OR}$ subtree (at position $par(p)$ in the lemma statement) with one of its immediate subtrees (p in the lemma statement).

Lemma 6. *Let $u \in L(\tau)$ be of size n and let a non-root position $p \in Pos(\tau)$ be such that $par(p) \in Pos_{\mathtt{OR}}(\tau)$. Consider*

$$\tau' := \tau[par(p) \leftarrow \tau_{|p}].$$

Then $\Upsilon^{\tau',par(p)}_{u}$ has a model if, and only if $\Upsilon^{\tau,p}_{u}$ has a model.

We can start the proof of Theorem 3.

$\Rightarrow$) Let $p \in \mathrm{Pos}(\tau)$, and suppose that p is useful for u in τ. We reason by induction on the depth of position p.

If $p = \epsilon$, then by Theorem 2, ϕ^{τ}_{u} has a model such that $x^{\epsilon}_{(1,n)}$ holds which entails the truthfulness of $\bigvee_{1 \leq i \leq j \leq n} x^{\epsilon}_{(i,j)}$. As a conlusion $\Upsilon^{\tau,\epsilon}_{u}$ has a model.

Otherwise, $p = par(p).i$ is defined and, by Definition 21, $par(p)$ is useful for u in τ. Now, let $\tau(par(p)) = \mathtt{OP}$, we distinguish the two cases $\mathtt{OP} \neq \mathtt{OR}$ and $\mathtt{OP} = \mathtt{OR}$.

- If $\mathtt{OP} \neq \mathtt{OR}$: since $par(p)$ is useful for u in τ, then by induction hypothesis, $\Upsilon^{\tau,par(p)}_{u}$ $(=\phi^{\tau}_{u} \wedge \bigwedge_{q \leq par(p)} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)})$ has a model, say ν. In particular, $\nu \models \bigvee_{1 \leq i \leq j \leq n} x^{par(p)}_{(i,j)}$; pick some $x^{par(p)}_{(i,j)} \in \nu$. Also, since $\nu \models \phi^{\tau}_{u}$, we have $\nu \models \psi^{\tau}_{u}$, which by Lemma 2 entails $\nu \models \Gamma^{\mathtt{OP},par(p)}_{i,j}$. In virtue of Definition 18 for the case $\mathtt{OP} \neq \mathtt{OR}$ and the fact $x^{par(p)}_{(i,j)} \in \nu$, we necessarily have some $x^{p}_{(i',j')} \in \nu$, which entails $\nu \models \phi^{\tau}_{u} \wedge \bigwedge_{q \leq p} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)} = \Upsilon^{\tau,p}_{u}$.
- Otherwise, $\mathtt{OP} = \mathtt{OR}$: by definition of p being useful with a $\mathtt{OR}$-parent position, we know that $u \in L(\tau[par(p) \leftarrow \tau_{|p}])$ and that $par(p)$ is useful for u in $\tau[par(p) \leftarrow \tau_{|p}]$. By induction hypothesis on $par(p)$[8], the formula $\Upsilon^{\tau[par(p) \leftarrow \tau_{|p}],p}_{u} = \phi^{\tau[par(p) \leftarrow \tau_{|p}]}_{u} \wedge \bigwedge_{q \leq par(p)} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)}$ has a model, and by Lemma 6, so does $\Upsilon^{\tau,p}_{u}$, which concludes.

$\Leftarrow$) Let $p \in \mathrm{Pos}(\tau)$ such that $\phi^{\tau}_{u} \wedge \bigwedge_{q \leq p} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)}$ has a model. We reason by induction on p.

If $p = \epsilon$, then we are done because by definition of useful positions, ϵ is always useful (provided $u \in L(\tau)$ which is an assumption of Theorem 3).

Otherwise, $p = par(p).i$ is defined.

We first establish that $par(p)$ is useful u in τ. Pick a model ν of $\Upsilon^{\tau,p}_{u}$. As formula $\bigwedge_{q \leq par(p)} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)}$ is weaker than formula $\bigwedge_{q \leq p} \bigvee_{1 \leq i \leq j \leq n} x^{q}_{(i,j)}$, we also have $\nu \models \Upsilon^{\tau,par(p)}_{u}$, which by induction hypothesis entails the usefulness of $par(p)$ for u in τ.

Now, consider $\mathtt{OP} = \tau(par(p))$, where the only interesting case is when $\mathtt{OP} = \mathtt{OR}$ (otherwise p is immediately useful for u in τ by Definition 21). Let $\tau' = \tau[par(p) \leftarrow \tau_{|p}]$.

By Lemma 6, because by assumption $\Upsilon^{\tau,p}_{u}$ has a model, so does $\Upsilon^{\tau',par(p)}_{u}$.

In particular, formula $\phi^{\tau'}_{u}$ has a model, which by Theorem 2 entails $u \in L(\tau')$. Also, by induction hypothesis on the depth of $par(p)$, we know that $par(p)$ is useful for u in τ'.

[8]Technically, the tree $\tau[par(p) \leftarrow \tau_{|p}]$ here is not τ, but notice that Theorem 2 holds for any tree, and that $par(p)$'s depth is strictly less that p's.

To sum up, We have shown that $par(p)$ is useful for u in τ, that $u \in L(\tau')$ and that $par(p)$ is useful for u in τ', so we can conclude by Definition 21 that p is useful for u in τ.

As announced, we next exploit our ability to characterize useful positions to guide the designer in a further developement of an attack tree.

D. Refining useful leaves

The way attack trees are designed relies on so-called refinement steps: starting from a leaf tree γ where the goal γ describes the global objective of the attack, the tree is refined by expanding the root position as some $\texttt{OP}(\gamma_1, \gamma_2)$. Then some of the new leaves, say at position 1, may be refined again into some $\texttt{OP}'(\gamma_{1.2}, \gamma_{2.2})$, and so on.

Remark that, after any refinement step, such a tree is conform to our Definition 7 of attack trees.

This refinement process is in most cases performed manually. The work [4] aims at providing a formal setting where a refinement step, that is a substitution of a leaf γ by a tree $\texttt{OP}(\gamma_1, \gamma_2)$, can be analyzed in the context of a given model $\mathcal{S}$ of the system. In this approach the central notion of the *Match* property[9](that may or may not hold for a refinement step) expresses a faithful rephrasing of goal γ as the combination of the subgoals γ_1 and γ_2. Otherwise stated, the tree obtained by a matching refinement, say τ', is such that the set of τ'-attacks in $\mathcal{S}$ coincides with the set of τ-attacks in $\mathcal{S}$.

Regarding quantitative analysis, it is clear that performing a matching refinement does not question the optimality of a previously computed witness (as described in the previous section). Nevertheless, as two new leaves have been generated, they might not be both useful. For a refinement involving either the `SAND` or the `AND` operator, all leaves of the refined tree are necessarily useful, whereas for a refinement involving the `OR` operator, it is then necessary to reconsider Theorem 3 on the two new leaves to determine their status.

A more critical situation happens when a non-matching refinement is considered: in such a case, there is more guarantee that the set of attacks in the system for the refined tree matches the set of attacks for the tree prior to this refinement, and the optimality of the witness may become obsolete. Hence, another round of quantitative analysis is needed to recompute a witness. Unless this new witness is identical to the previous one, the whole process to determine useful positions needs being restarted on the entire tree because previously useless leaves might become useful for this updated witness.

VII. Conclusion

In this paper, we have developed a methodology to guide a security expert in the design of an attack tree for a given system. In contrast to standard approaches where attack trees are labeled with actions that an attacker needs to execute, we consider attack trees where leaves represent reachability goals in the analyzed system. Such attack trees are homogeneous throughout the entire design process: at the earliest stage it might be a single node labeled with the attacker's goal which is then refined according to the system and a new attack tree with goal-labeled nodes is obtained. The model of attack trees with reachability goals enables an early-stage analysis, i.e., the analysis with trees that have not yet been fully refined.

To formalize our attack trees and be able to express the link with the analyzed model, we defined trace semantics for attack trees. It required us to introduce the operations of enhanced concatenation and shuffle of regular languages which are not classic in the standard theory of formal languages but are well chosen to reflect the path semantics of attack trees with reachability goals, as introduced in [4]. When the τ-monitoring of a system $\mathcal{S}$ is considered, the "winning" paths in the monitored system $\tau[\mathcal{S}]$, i.e., the paths in $\mathcal{S}$ that have their trace embedded in the language of attack tree τ, correspond exactly to the paths in the path semantics of [4]. Under the path semantics, an attack tree is a combination of reachability goals and the enhanced operators correspond to the `SAND` and `AND` refinements, as defined in [4]. It is to be noticed that in this path-based setting, `AND` should not be seen as a "true parallel", but rather as a conjunctive operator that allows for any superposition of subgoals in an execution.

Our ability to monitor a transition system with an attack tree is easily extended to quantitative systems. This enables quantitative analysis, i.e., giving a value to the tree, and synthesizing an optimal path. We discussed it on the example of priced timed automata (PTA) because it is a rich model, but our methodology applies to any quantitative system for which there exists an algorithm synthesizing optimal paths. Examples of such systems are multi-priced timed automata with single cost minimization and bounds on other costs [20], but also all sorts of weighted automata [23], [22].

We consider a single optimal path that defines the value assigned to the tree and guides the tree creation. In this context, the guided construction of attack tree is correct, in the sense that if a position is suggested to be further refined, it does correspond to an optimal path in the system.

Classically, an attack tree represents a collection of potential attacks. However, in this paper we suggest to refine an attack tree only with an optimal attack in mind. So a natural question that arises is how an expert who wants to cover more (or even all) attacks, rather than only optimal ones, should proceed? Query for attacks that are a bit less than optimal, but synthesizing all attacks, or at least all elementary attacks (i.e., without loops) is a much harder problem. A thorough investigation should be conducted to estimate how hard this problem is in practice. In the mean time, we may suggest a procedure which is standard in practical risk analysis: starting to analyze an optimal attack, then mitigate it by patching the system accordingly, and then search for optimal attacks in the updated system.

In the future, we would also like to study the complexity of the problems underlying our framework. In particular, the problem of optimal reachability clearly depends on the quantitative model that is used. It is known to be EXPTIME-complete for PTAs [2], but investigating its complexity for

[9]There are also some weaker notions that are less relevant here.

other types of systems would provide valuable information about the practical applicability of our framework. Similarly, deciding if a position is useful is in NP, as it can be solved with the help of the SAT problem. However, we have not yet identified its lower bound. Since the conjunctive normal form of the usefulness-test formula is not a Horn formula, we cannot immediately deduce that the problem is polynomial, and a different reasoning would need to be used.

Finally, we have defined the notion of useless position with respect to one optimal trace. In other words, in the current framework, a position is useless if it does not contribute to the optimal trace under consideration. However, there is no guarantee that there is no other optimal trace where this position would be useful. It would thus be interesting to redefine the notion of uselessness of a position with respect to all optimal traces. Proving that a position is useless in this new sense would be of course more difficult, but such positions could simply be removed from the tree without losing any relevant attacks.

Acknowledgment: This research was supported by the Direction Générale de l'Armement (DGA) - Ministère de la Défense, France.

References

[1] Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? In *ATVA*, volume 11, pages 482–491. Springer, 2011.

[2] Rajeev Alur, Mikhail Bernadsky, and Parthasarathy Madhusudan. Optimal reachability for weighted timed games. In *ICALP*, pages 122–133. Springer, 2004.

[3] Rajeev Alur and David L Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.

[4] Maxime Audinot, Sophie Pinchinat, and Barbara Kordy. Is my attack tree correct? In *ESORICS (1)*, volume 10492 of *LNCS*, pages 83–102. Springer, 2017.

[5] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of model checking*. MIT press, 2008.

[6] Jiří Barnat, Luboš Brim, Vojtěch Havel, Jan Havlíček, Jan Kriho, Milan Lenčo, Petr Ročkai, Vladimír Štill, and Jiří Weiser. Divine 3.0–an explicit-state model checker for multithreaded c & c++ programs. In *CAV*, pages 863–868. Springer, 2013.

[7] Gerd Behrmann, Kim G Larsen, and Jacob I Rasmussen. Priced timed automata: Algorithms and applications. In *FMCO*, volume 3657, pages 162–182. Springer, 2004.

[8] Peter Bezděk, Nikola Beneš, Jiří Barnat, and Ivana Černá. Ltl parameter synthesis of parametric timed automata. In *SEFM*, pages 172–187. Springer, 2016.

[9] Franziska Biegler, Mark Daley, and Ian McQuillan. Algorithmic decomposition of shuffle on words. *TCS*, 454:38–50, 2012.

[10] EAC Advisory Board and Standards Board. Election Operations Assessment – Threat Trees and Matrices and Threat Instance Risk Analyzer (TIRA). https://www.eac.gov/assets/1/28/Election_Operations_Assessment_Threat_Trees_and_Matrices_and_Threat_Instance_Risk_Analyzer_(TIRA).pdf, 2009.

[11] Olga Gadyatskaya, René Rydhof Hansen, Kim Guldstrand Larsen, Axel Legay, Mads Chr. Olesen, and Danny Bøgsted Poulsen. Modelling Attack-defense Trees Using Timed Automata. In *FORMATS*, volume 9884 of *LNCS*, pages 35–50. Springer, 2016.

[12] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Automata theory, languages, and computation. *International Edition*, 24, 2006.

[13] Joanna Jedrzejowicz and Andrzej Szepietowski. Shuffle languages are in p. *TCS*, 250(1-2):31–53, 2001.

[14] Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Rolando Trujillo-Rasua. Attack Trees with Sequential Conjunction. In *SEC*, volume 455 of *IFIP AICT*, pages 339–353. Springer, 2015.

[15] Aivo Jürgenson and Jan Willemson. Computing exact outcomes of multi-parameter attack trees. In *OTM Conferences (2)*, volume 5332 of *LNCS*, pages 1036–1051. Springer, 2008.

[16] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. Adtool: security analysis with attack–defense trees. In *International Conference on Quantitative Evaluation of Systems*, pages 173–176. Springer, 2013.

[17] Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Attack–defense trees. *J. Log. Comput.*, 24(1):55–87, 2014.

[18] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *CSR*, 13–14(0):1–38, 2014.

[19] Rajesh Kumar, Enno Ruijters, and Mariëlle Stoelinga. Quantitative Attack Tree Analysis via Priced Timed Automata. In *FORMATS*, volume 9268 of *LNCS*, pages 156–171. Springer, 2015.

[20] Kim Guldstrand Larsen and Jacob Illum Rasmussen. Optimal conditional reachability for multi-priced timed automata. In *International Conference on Foundations of Software Science and Computation Structures*, pages 234–249. Springer, 2005.

[21] Sjouke Mauw and Martijn Oostdijk. Foundations of Attack Trees. In *ICISC*, volume 3935 of *LNCS*, pages 186–198. Springer, 2005.

[22] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *JALC*, 7(3):321–350, 2002.

[23] Mehryar Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

[24] National Electric Sector Cybersecurity Organization Resource (NESCOR). Analysis of Selected Electric Sector High Risk Failure Scenarios. https://www.nevermoresecurity.com/wp-content/uploads/2017/10/Analysis-of-Selected-Electric-Sector-High-Risk-Failure-Scenarios-%E2%80%93-Version-2.0-.pdf, 2015.

[25] Peter Niebert, Stavros Tripakis, and Sergio Yovine. Minimum-time reachability for timed automata. In *IEEE Mediteranean Control Conference*. Citeseer, 2000.

[26] Bruce Schneier. Attack Trees: Modeling Security Threats. *Dr. Dobb's Journal of Software Tools*, 24(12):21–29, 1999.

[27] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.

[28] Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[29] Uppaal cora. http://people.cs.aau.dk/ãdavid/cora/, 2010.

Appendix A
Proof of Proposition 2

1) If $u \in L(\mathcal{A}_1 \odot \mathcal{A}_2)$, then there exists a run of $\mathcal{A}_1 \odot \mathcal{A}_2$ accepting u. Let $q_0, \ldots, q_n$ be such an accepting run. For all $i' \in [0, n]$, we have either $q_{i'} \in Q_1$ or $q_{i'} \in Q_2$. Additionally, we have $q_0 = q_1^0 \in Q_1$ and $q_n \in F_2 \subseteq Q_2$. Let $i \in [0, n-1]$ be such that $q_i \in Q_1$ and $q_{i+1} \in Q_2$. By definition of δ, for all $i' \in [0, n-1]$, if $q_{i'} \in Q_2$ then $q_{i'+1} \in Q_2$, so i is uniquely defined. We have that (1) for all $i' \in [0, i-1]$, it holds that $q_{i'} \in Q_1$ and $q_{i'+1} \in \delta_1(q_{i'}, u(i'+1))$, (2) either $q_i \in F_1$, or $u(i+1) = b' \wedge b''$ and $\delta_1(q_i, b') \cap F_1 \neq \emptyset$ and $q_{i+1} \in \delta_2(q_2^0, b'')$, and (3) for all $i' \in [i+1, n-1]$, $q_{i'} \in Q_2$ and $q_{i'+1} \in \delta_2(q_{i'}, u(i'+1))$. Now we distinguish between the two cases of (2).

- Case $q_i \in F_1$: By definition of δ, we have that $q_{i+1} \in \delta_2(q_2^0, u(i+1))$. Let $u_1 = u(0) \ldots u(i)$ and $u_2 = u(i+1) \ldots u(n)$. We have that $q_0 = q_1^0$, and $q_i \in F_1$, and we have (1), so $q_0, \ldots, q_i$ is a run of $\mathcal{A}_1$ accepting u_1. We also have that $q_{i+1} \in \delta_2(q_2^0, u(i+1))$, and (3), and $q_n \in F_2$, so $q_2^0, q_{i+1}, \ldots, q_n$ is a run in $\mathcal{A}_2$ accepting u_2. As $u = u_1.u_2$, we have that $u \in L(\mathcal{A}_1) \odot L(\mathcal{A}_2)$
- Case $u(i+1) = b' \wedge b''$ and $\delta_1(q_i, b') \cap F_1 \neq \emptyset$ and $q_{i+1} \in \delta_2(q_2^0, b'')$: Let $u_1 = u(0) \ldots u(i)$ and $u_2 = u(i+$

2)$\dots u(n)$. Let $q_1^f \in \delta_1(q_i, b') \cap F_1$. By (1) and $q_0 = q_1^0$, we have that $q_0, \dots, q_i, q_1^f$ is a run of $\mathcal{A}_1$ accepting $u_1 b'$. By (3) and $q_{i+1} \in \delta_2(q_2^0, b'')$ and $q_n \in F_2$, we have that $q_2^0, q_{i+1}, \dots, q_n$ is a run of $\mathcal{A}_2$ accepting $b'' u_2$. As $u = u_1.(b' \wedge b'').u_2$ with $u_1 b' \in L(\mathcal{A}_1)$ and $b'' u_2 \in L(\mathcal{A}_2)$, we have that $u \in L(\mathcal{A}_1) \odot L(\mathcal{A}_2)$

If $u \in L(\mathcal{A}_1) \odot L(\mathcal{A}_2)$, then there exist two possible cases: either $u = u_1.u_2$ with $u_i \in L(\mathcal{A}_i)$, or $u = u_1.(b' \wedge b'').u_2$ with $u_1 b' \in L(\mathcal{A}_1)$ and $b'' u_2 \in L(\mathcal{A}_2)$.

- Suppose that $u = u_1.u_2$, with $u_i \in L(\mathcal{A}_i)$. Then, let $q_0, \dots, q_i$ be a run of $\mathcal{A}_1$ accepting u_1, and let $q_2^0, q_{i+1}, \dots, q_n$ be a run of $\mathcal{A}_2$ accepting u_2. For all $i' \in [0, i]$, we have $q_{i'} \in Q_1$ and $q_{i'+1} \in \delta_1(q_{i'}, u(i'+1))$, so $q_{i'+1} \in \delta(q_{i'}, u(i'+1))$. Since $q_i \in F_1$ and $q_{i+1} \in \delta_2(q_2^0, u(i+1))$, we have $q_{i+1} \in \delta(q_i, u(i+1))$. Additionally, for all $i' \in [i+1, n-1]$, we have $q_{i'} \in Q_2$ and $q_{i'+1} \in \delta_2(q_{i'}, u(i'+1))$, so $q_{i'+1} \in \delta(q_{i'}, u(i'+1))$. Finally, we also have $q_0 = q_1^0$ and $q_n \in F_2$, thus $q_0, \dots, q_n$ is a run of $\mathcal{A}_1 \odot \mathcal{A}_2$ accepting u, and $u \in L(\mathcal{A}_1 \odot \mathcal{A}_2)$.
- Suppose that $u = u_1.(b' \wedge b'').u_2$, with $u_1 b' \in L(\mathcal{A}_1)$ and $b'' u_2 \in L(\mathcal{A}_2)$. Let $q_0, \dots, q_i, q_1^f$ be a run of $\mathcal{A}_1$ accepting $u_1 b'$, and let $q_2^0, q_{i+1}, \dots, q_n$ be a run in $\mathcal{A}_2$ accepting $b'' u_2$. For all $i' \in [0, i-1]$, we have $q_{i'} \in Q_1$ and $q_{i'+1} \in \delta_1(q_{i'}, u(i'+1))$, so $q_{i'+1} \in \delta(q_{i'}, u(i'+1))$. We have that $q_1^f \in \delta_1(q_i, b')$, and $q_{i+1} \in \delta_2(q_2^0, b'')$, so $q_{i+1} \in \delta(q_i, b' \wedge b'')$. Additionally, for all $i' \in [i+1, n-1]$, we have $q_{i'} \in Q_2$ and $q_{i'+1} \in \delta_2(q_{i'}, u(i'+1))$, thus $q_{i'+1} \in \delta(q_{i'}, u(i'+1))$. Finally, we also have $q_0 = q_1^0$ and $q_n \in F_2$, which implies that $q_0, \dots, q_n$ is a run of $\mathcal{A}_1 \odot \mathcal{A}_2$ accepting u, and thus $u \in L(\mathcal{A}_1 \odot \mathcal{A}_2)$.

2) First, remark that Definition 3 can be stated differently, in a way more convenient to proving Proposition 2.

Lemma 7. *Let $u, v \in C^+(X)^*$. We have $w \in u \otimes v$ if, and only if, there exist two sets J and K such that $J \cup K = [1, |w|]$, and for all $i \in [1, |w|]$,*

$$w(i) = \begin{cases} u(j_i) \text{ if } i \in J \setminus K \\ v(k_i) \text{ if } i \in K \setminus J \\ u(j_i) \wedge v(k_i) \text{ otherwise, thus } i \in J \cap K \end{cases}$$

where the increasing sequences $(j_i)_{i \in [1,|w|]}$ and $(k_i)_{i \in [1,|w|]}$ are defined by recurrence over i as follows.
First $j_1 = k_1 = 1$, then for all $i \in [2, |w|]$,

$$j_i = \begin{cases} j_{i-1} + 1 \text{ if } i-1 \in J \\ j_{i-1} \text{ otherwise} \end{cases} \quad k_i = \begin{cases} k_{i-1} + 1 \text{ if } i-1 \in K \\ k_{i-1} \text{ otherwise} \end{cases}$$

We do not prove this lemma, but for example we have seen that $b_1 b_2 \otimes b_3 = \{b_1 b_2 b_3, b_1(b_2 \wedge b_3), b_1 b_3 b_2, (b_1 \wedge b_3) b_2, b_3 b_1 b_2\}$. Take $b_1(b_2 \wedge b_3) \in b_1 b_2 \otimes b_3$; the sets J and K in Lemma 7 are respectively $\{1, 2\}$ and $\{2\}$, while for $b_3 b_1 b_2$ the sets are $\{2, 3\}$ and $\{1\}$.

Let $u \in L(\mathcal{A}_1) \otimes L(\mathcal{A}_2)$. Let $u = v \otimes w$ such that $v \in L(\mathcal{A}_1)$ and $w \in L(\mathcal{A}_2)$. Let $q_0, \dots, q_{m+1}$ be a run in $\mathcal{A}_1$ accepting v, and let $q'_0, \dots, q'_{m'+1}$ be a run in $\mathcal{A}_2$ accepting w. Let $n = |u|$.

Let J and K be the two sets defined in Lemma 7, and let $(j_i)_{i \in [1,n]}$ and $(k_i)_{i \in [1,n]}$ be the two sequences defined in Lemma 7. We also define by convention $j_0 = k_0 = 0$. We now show that the sequence $(q_{j_0}, q'_{k_0}), (q_{j_1}, q'_{k_1}), \dots, (q_{j_n}, q'_{k_n})$ is a run in $\mathcal{A}_1 \otimes \mathcal{A}_2$ that accepts u. As $q_0, \dots, q_m$ is a run in $\mathcal{A}_1$ accepting v, we have that $q_0 = q_1^0$ and $q_m \in F_1$, and with a similar reasoning, we have $q'_0 = q_2^0$ and $q'_{m'+1} \in F_2$. Moreover, for $2 \le i \le n$, we have three possible exclusive cases

- $i - 1 \in J$ and $i - 1 \notin K$: we have $j_i = j_{i-1} + 1$, and as $q_0, \dots, q_m$ is a run in $\mathcal{A}_1$ accepting v, we have that $q_{j_i} \in \delta_1(q_{j_{i-1}}, v(j_i))$. Besides, we have $k_i = k_{i-1}$ so immediately $q'_{k_i} = q'_{k_{i-1}}$. By the definition of the transition function of $\mathcal{A}_1 \otimes \mathcal{A}_2$, and as $v(j_i) = u(i)$ we have that $(q_{j_i}, q'_{k_i}) \in \delta((q_{j_{i-1}}, q'_{k_{i-1}}), u(i))$.
- $i - 1 \notin J$ and $i - 1 \in K$: we have $j_i = j_{i-1}$, so immediately $q_{j_i} = q_{j_{i-1}}$. Besides, we have $k_i = k_{i-1} + 1$, and as $q'_0, \dots, q'_{m'}$ is a run in $\mathcal{A}_2$ accepting w, we have that $q'_{k_i} \in \delta_2(q'_{k_{i-1}}, w(k_{i-1}))$. By the definition of the transition function of $\mathcal{A}_1 \otimes \mathcal{A}_2$, and as $w(k_i) = u(i)$ we have that $(q_{j_i}, q'_{k_i}) \in \delta((q_{j_{i-1}}, q'_{k_{i-1}}), u(i))$.
- $i - 1 \in J$ and $i - 1 \in K$: we have $j_i = j_{i-1} + 1$, and as $q_0, \dots, q_m$ is a run in $\mathcal{A}_1$ accepting v, we have that $q_{j_i} \in \delta_1(q_{j_{i-1}}, v(j_i))$. Besides, we have $k_i = k_{i-1} + 1$, and as $q'_0, \dots, q'_{m'}$ is a run in $\mathcal{A}_2$ accepting w, we have that $q'_{k_i} \in \delta_2(q'_{k_{i-1}}, w(k_i))$. We also have that $u(i) = v(j_i) \wedge w(k_i)$. By the definition of the transition function of $\mathcal{A}_1 \otimes \mathcal{A}_2$, we have that $(q_{j_i}, q'_{k_i}) \in \delta((q_{j_{i-1}}, q'_{k_{i-1}}), u(i))$.

In any case, we have that $(q_{j_i}, q'_{k_i}) \in \delta((q_{j_{i-1}}, q'_{k_{i-1}}), u(i))$. As additionally $(q_{j_0}, q'_{k_0}) = (q_1^0, q_2^0)$ and $(q_{j_n}, q'_{k_n}) \in F_1 \times F_2$, we conclude that $(q_{j_0}, q'_{k_0}), \dots, (q_{j_n}, q'_{k_n})$ is a run in $\mathcal{A}_1 \otimes \mathcal{A}_2$ accepting u.

Now we turn to the other direction of the inclusion. Let $u \in L(\mathcal{A}_1 \otimes \mathcal{A}_2)$. Let $(q_0, q'_0), \dots, (q_n, q'_n)$ be a run in $\mathcal{A}_1 \otimes \mathcal{A}_2$ accepting u. By definition of δ, for each $i \in [0, n-1]$ we have 3 possible case: (1) $q_{i+1} = q_i$ and $q'_{i+1} \in \delta_2(q'_i, u(i+1))$ or (2) $q_{i+1} \in \delta_1(q_i, u(i+1))$ and $q'_{i+1} = q'_i$ or (3) $u(i+1) = b' \wedge b''$ and $q_{i+1} \in \delta_1(q_i, b')$ and $q'_{i+1} \in \delta_2(q'_i, b'')$. We define the two set J and K such that for all $i \in [0, n-1]$, $i+1 \in J$ if and only if case (2) or (3) hold for i, and $i+1 \in K$ if and only if case (1) or (3) hold for i. Necessarily $J \cup K = [0, n-1]$. Remark that if for some i, case (1) and (2) both hold, then case (3) holds as well, so for all $i+1 \in J \cap K$, case (3) holds for i. Moreover, for all $i+1 \in J \setminus K$, case (2) holds for i but not case (1) and case (3) , and for all $i+1 \in K \setminus J$, case (1) holds for i but not case (2) and case (3). We define the two sequences of pairs of integers $j_1, \dots, j_n$ and $k_1, \dots, k_n$ by recurrence such that $j_1 = k_1 = 1$, and for all $2 \le i \le n$,

$$j_i = \begin{cases} j_{i-1} + 1 \text{ if } i-1 \in J \\ j_{i-1} \text{ otherwise} \end{cases} \quad k_i = \begin{cases} k_{i-1} + 1 \text{ if } i-1 \in K \\ k_{i-1} \text{ otherwise} \end{cases}$$

Now we define v and w to be the two words such that for all $1 \le i \le n$, $u(i) = \begin{cases} v(j_{i-1}) \text{ if } i \in J \setminus K \\ w(k_{i-1}) \text{ if } i \in K \setminus J \\ v(j_{i-1}) \wedge w(k_{i-1}) \text{ if } i \in J \cap K \end{cases}$

By Lemma 7, we have that $u \in v \otimes w$. We conclude by showing that $v \in L(\mathcal{A}_1)$ and $w \in L(\mathcal{A}_2)$.

For all $i \in J$, case (2) or (3) holds, which means that either $u(i+1) = v(j_{i+1})$ and $q_{i+2} \in \delta_1(q_{i+1}, u(i+1))$ or $u(i+1) = v(j_{i+1}) \wedge w(k_{i+1})$ $q_{i+2} \in \delta_1(q_{i+1}, v(j_{i+1}))$, so for all $i \in J$, we have $q_{i+2} \in \delta_1(q_{i+1}, v(j_{i+1}))$. Additionally, we have $q_0 = q_1^0$ and $q_n \in F_1$. Also, remark that for all $i \in [1,n] \backslash J$, $q_{i+1} = q_{j+1}$ where $j = \min\{j \in J \cup \{n\} \mid j > i\}$. The sequence $(q_{j-1})_{j \in J}, q_n$ is a run in $\mathcal{A}_1$ accepting v. Similarity, for all $i \in K$, case (1) or (3) holds, which means that either $u(i) = v(k_{i+1})$ and $q'_{i+2} \in \delta_2(q'_{i+1}, u(i+1))$ or $u(i+1) = v(j_{i+1}) \wedge w(k_{i+1})$ $q'_{i+2} \in \delta_2(q'_{i+1}, w(k_{i+1}))$, so for all $i \in K$, we have $q'_{i+2} \in \delta_2(q'_{i+1}, w(k_{i+1}))$. Additionally, we have $q'_0 = q_2^0$ and $q'_n \in F_2$. Also, remark that for all $i \in [1,n] \setminus K$, $q_{i+1} = q_{k+1}$ where $k = \min\{k \in K \cup \{n\} \mid k > i\}$. The sequence $(q'_{k-1})_{k \in K}, q'_n$ is a run in $\mathcal{A}_2$ accepting w

Appendix B
Proof of Lemma 6

Recall the lemma statement: Let $u \in L(\tau)$ of size n and let a non-root position $p \in \mathrm{Pos}(\tau)$ be such that $par(p) \in \mathrm{Pos}_{\mathtt{OR}}(\tau)$. Consider $\tau' := \tau[par(p) \leftarrow \tau_{|p}]$. Then
$\Upsilon_u^{\tau', par(p)}$ has a model if, and only if $\Upsilon_u^{\tau,p}$ has a model.

We make a proof only for the case $p = par(p).1$, as the other case $p = par(p).2$ is symmetric.

First of all, remark that τ and τ' are very similar: they differ only by the parts below position $par(p)$, and the two subtrees $\tau_{|p}$ and $\tau'_{|par(p)}$ are equal. We make use of this remark to separate the formulas $\Upsilon_u^{\tau', par(p)}$ and $\Upsilon_u^{\tau,p}$, into a conjunct of respectively three and six subformulas.

To implement this separation, we first use Lemma 2 to flatten $\Upsilon_u^{\tau', par(p)}$ into a conjunction, and then separate this conjunction into three subformulas, each subformula being characterized by the relation between $par(p)$ and the positions the subformula refers to. There are three kinds of sets of positions relative to $par(p)$: the first kind is when all positions are below $par(p)$, the second kind is when no positions are below $par(p)$, and the third kind when some position are below $par(p)$ and some others are not. Let ϕ_{out}, ϕ_{in} and ϕ_{inter} be the three separated subformulas, so that ϕ_{out} is the conjunction of all formulas of the first kind, ϕ_{out} is the conjunction of all formulas of the second kind, and ϕ_{inter} is the conjunction of all formulas of the third kind. It can be shown that

- $\phi_{\text{in}} = \bigwedge_{\substack{1 \le i \le j \le n, \\ q \in \ell(\tau), \\ par(p) \le q, \\ u(j) \not\models \tau(q)}} \neg x^q_{(i',j')} \wedge \bigwedge_{\substack{q \in \mathrm{Pos}(\tau) \\ \backslash \ell(\tau), \\ par(p) \le q}} \Gamma_{1,n}^{\tau(q),q} \wedge \bigvee_{1 \le i \le j \le n} x^{par(p)}_{(i,j)}$
- $\phi_{\text{inter}} = \bigwedge_{1 \le i \le j \le n} \Gamma_{i,j}^{\tau(par(par(p))), par(par(p))}$
- $\phi_{\text{out}} = \bigwedge_{\substack{1 \le i \le j \le n, \\ q \in \ell(\tau), \\ par(par(p)) \not\le q, \\ u(j) \not\models \tau(q)}} \neg x^q_{(i',j')} \wedge \bigwedge_{\substack{q \in \mathrm{Pos}(\tau) \\ \backslash \ell(\tau), \\ par(par(p)) \not\le q}} \Gamma_{1,n}^{\tau(q),q} \wedge \bigwedge_{q \le par(par(p))} \bigvee_{1 \le i \le j \le n} x^q_{(i,j)}$

Similarly, we flatten $\Upsilon_u^{\tau,p}$ and separate the flattened version the same way into three subformulas. But this time, it can be shown that the subformula of the first kind is

$$\phi_{\text{in}}{\downarrow_1} \wedge \psi_u^{\tau_{|par(p).2}}{\downarrow_{par(p).2}} \wedge \Gamma_{1,n}^{\tau(par(p)), par(p)} \wedge \bigvee_{1 \le i \le j \le n} x^{par(p)}_{(i,j)},$$

the subformula of the second kind is ϕ_{inter}, and the subformula of the third kind is ϕ_{out}.

To recap the separation of $\Upsilon_u^{\tau', par(p)}$ and $\Upsilon_u^{\tau,p}$, we can rewrite

- $\Upsilon_u^{\tau', par(p)}$ as $\phi_{\text{in}} \wedge \phi_{\text{inter}} \wedge \phi_{\text{out}}$, and
- $\Upsilon_u^{\tau,p}$ as $\phi_{\text{in}}{\downarrow_1} \wedge \psi_u^{\tau_{|par(p).2}}{\downarrow_{par(p).2}} \wedge \Gamma_{1,n}^{\tau(par(p)), par(p)} \wedge \bigvee_{1 \le i \le j \le n} x^{par(p)}_{(i,j)} \wedge \phi_{\text{inter}} \wedge \phi_{\text{out}}$

$\Rightarrow$): Suppose that $\nu \models \Upsilon_u^{\tau', par(p)}$.

We derive two valuations ν_{out} and ν_{in} that restrict ν to particular domains:

- $dom(\nu_{\text{in}}) = dom(\nu) \cap \{x^q_{(i,j)} \mid q \in \mathrm{Pos}(\tau[par(p) \leftarrow \tau_{|p}])$ with $par(p) \le q, i \le j \in \mathbb{N}\}$, and
- $dom(\nu_{\text{out}}) = dom(\nu) \cap \{x^q_{(i,j)} \mid q \in \mathrm{Pos}(\tau[par(p) \leftarrow \tau_{|p}])$ with $par(p) \not\le q, i \le j \in \mathbb{N}\}$.

It is easy to establish that $dom(\nu_{\text{in}}) \subseteq var(\phi_{\text{in}})$, and $dom(\nu_{\text{out}}) \subseteq var(\phi_{\text{out}})$.

Let ν'_{in} be the valuation of domain $\{x^q_{(i,j)} \mid q \in \mathrm{Pos}(\tau)$ with $par(p) \le q, 1 \le i \le j \le n\}$ defined by $\nu'_{\text{in}}(x^q_{(i,j)}) = \mathtt{tt}$ iff $q = par(p)$ or $q = p.q'$ and $\nu_{\text{in}}(x^{par(p).q'}_{(i,j)}) = \mathtt{tt}$. Clearly, $dom(\nu'_{\text{in}}) \cap dom(\nu_{\text{out}}) = \emptyset$.

We have $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{inter}}$ because $\nu \models \phi_{\text{inter}}$ and $\nu'_{\text{in}} \cup \nu_{\text{out}}$ and ν agree on $var(\phi_{\text{inter}})$.

Now, the statement "$\nu'_{\text{in}} \models x^{par(p).1.q}_{(i,j)}$ iff $\nu_{\text{in}} \models x^{par(p).q}_{(i,j)}$ for all $par(p).q \in dom(\nu_{\text{in}})$ and every i, j" implies the statement "$\nu'_{\text{in}} \models \phi_{\text{in}}{\downarrow_1}$ iff $\nu_{\text{in}} \models \phi_{\text{in}}$".

Additionally, $\emptyset \models \psi_u^{\tau_{|par(p).2}}{\downarrow_{par(p).2}} \wedge \Gamma_{1,n}^{\tau(par(p)), par(p)}$ and $dom(\nu'_{\text{in}}) \cap var(\phi_{\text{out}}) = dom(\nu_{\text{out}}) \cap var(\phi_{\text{in}}) = \emptyset$ and $\nu_{\text{out}} \models \phi_{\text{out}}$, so by Lemma 1, $\nu'_{\text{in}} \models \phi_{\text{in}}{\downarrow_1} \wedge \psi_u^{\tau_{|par(p).2}}{\downarrow_{par(p).2}} \wedge \Gamma_{1,n}^{\tau(par(p)), par(p)}$ and $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{in}}{\downarrow_1} \wedge \psi_u^{\tau_{|par(p).2}}{\downarrow_{par(p).2}} \wedge \Gamma_{1,n}^{\tau(par(p)), par(p)} \wedge \phi_{\text{out}}$. As Additionally $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{inter}}$ and $\nu_{\text{out}} \models \bigvee_{1 \le i \le j \le n} x^{par(p)}_{(i,j)}$, we have $\nu_{\text{in}} \cup \nu_{\text{out}} \models \phi_u^{\tau} \wedge \bigwedge_{q \le p} \bigvee_{1 \le i \le j \le n} x^q_{(i,j)} = \Upsilon_u^{\tau,p}$.

$\Leftarrow$): Suppose that $\nu \models \Upsilon_u^{\tau,p}$.

We let ν_{out} and ν_{in} restrict ν to

- $dom(\nu_{\text{out}}) = dom(\nu) \cap \{x^q_{(i,j)} \mid q \in \mathrm{Pos}(\tau)$ with $par(p) \not\le q, i \le j \in \mathbb{N}\}$, and
- $dom(\nu_{\text{in}}) = dom(\nu) \cap \{x^q_{(i,j)} \mid q \in \mathrm{Pos}(\tau)$ with $q = par(p)$ or $p \le q, i \le j \in \mathbb{N}\}$.

It can be estblished that $dom(\nu_{\text{out}}) \subseteq var(\phi_{\text{out}})$ and $dom(\nu_{\text{in}}) \subseteq var(\phi_{\text{in}}{\downarrow_1}) \cup \{x^{par(p)}_{(i,j)} \mid i \le j \in \mathbb{N}\}$.

Let ν'_{in} be the valuation of domain $\{x^q_{(i,j)} \mid q \in \mathrm{Pos}(\tau[par(p) \leftarrow \tau_{|p}])$ with $par(p) \le q, 1 \le i \le j \le n\}$ defined by $\nu'_{\text{in}}(x^q_{(i,j)}) = \mathtt{tt}$ iff $q = par(p).q'$ and $\nu_{\text{in}}(x^{p.q'}_{(i,j)}) = \mathtt{tt}$. Clearly, $dom(\nu'_{\text{in}}) \cap dom(\nu_{\text{out}}) = \emptyset$.

Now, $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{inter}}$ because $\nu \models \phi_{\text{inter}}$ and $\nu'_{\text{in}} \cup \nu_{\text{out}}$ and ν agree on $var(\phi_{\text{inter}})$.

The statement "$\nu'_{\text{in}} \models x^{par(p).q}_{(i,j)}$ iff $\nu_{\text{in}} \models x^{par(p).1.q}_{(i,j)}$ for all $par(p).1.q \in dom(\nu_{\text{in}})$ and every i, j" implies the statement "$\nu'_{\text{in}} \models \phi_{\text{in}}$ iff $\nu_{\text{in}} \models \phi_{\text{in}}{\downarrow_1}$". Finally, $dom(\nu'_{\text{in}}) \cap var(\phi_{\text{out}}) = dom(\nu_{\text{out}}) \cap var(\phi_{\text{in}}) = \emptyset$, so by Lemma 1, $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{in}} \wedge \phi_{\text{out}}$, and as $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{inter}}$, we have $\nu'_{\text{in}} \cup \nu_{\text{out}} \models \phi_{\text{in}} \wedge \phi_{\text{out}} \wedge \phi_{\text{inter}} = \Upsilon_u^{\tau', par(p)}$, which concludes.