Efficient attack–defense tree analysis using Pareto attribute domains

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

Wojciech Wideł
Univ Rennes, INSA Rennes,
CNRS, IRISA, Rennes, France
wojciech.widel@irisa.fr

***Abstract*—The cheapest attacks are often time-consuming, and those requiring high level of technical skills might occur rarely but result in disastrous consequences. Therefore, analysis focusing on a single parameter at a time, e.g., only cost or time, is insufficient for the successful selection of the appropriate measures increasing system's security. In practice, security engineers are thus confronted with the problem of multi-parameter analysis.**

The objective of this work is to address this problem and propose a sound, general framework for multi-parameter analysis of security. In order to ensure the usability of our solution for real-life applications, our proposal relies on the attack–defense tree model that security experts from industry are already familiar with. We present mathematical foundations of our framework and characterize the class of parameters it is suitable for. We identify conditions under which the proposed method applies to attack–defense trees where several nodes represent the same action. We discuss the complexity of our approach and implement the underlying algorithms in a proof of concept tool. We analyze its performance on a number of trees of varying complexity, and validate our proposal on a case study borrowed from industry.

***Index Terms*—Attack tree, attack–defense tree, multi-objective optimization, Pareto frontier**

I. Introduction

Performing quantitative analysis of security is one of the primary tasks of security or risk analysis experts. Depending on the analyzed system, its purpose as well as the underlying security requirements and the attacker's profile, different parameters (also called metrics or attributes) may be considered. *Cost*, *time* or *probability* of attacking, *expected damage* or *impact* that an attack would cause to the system, or the *investment* necessary to protect the system are examples of such security-relevant parameters. Estimating the value of these parameters helps the expert to evaluate the degree of security of the system, identify its most vulnerable components, and decide which attacks should be prevented with the highest priority.

Computing the value of the individual parameters is not difficult, but is often not sufficient. How to assess an attack having a very low *probability of success* but whose *impact* might be disastrous for the system? Which amongst the parameters *cost*, *time*, or *difficulty level* should the expert consider first to identify the most probable strategy that an attacker having limited resources (in terms of money and time) and technical skills would follow?

The above issues motivate the need for a multi-parameter analysis, i.e., an analysis where several parameters are taken into account at the same time. There are two main approaches for addressing such analysis. The first one consists in combining all relevant parameters into a new metrics. A good example here is the *risk* metrics, relying on the *cost*, *probability*, and *impact* parameters. One may also think about translating all parameters of interest into the same scale (e.g., monetary cost) and then performing a single-parameter analysis. This solution is however cumbersome, since numerous classical parameters (e.g., *difficulty*, *technical skill's level*) work with ordinal scales, such as low–medium–high, and might not be easily translatable into the numerical ones.

The second approach for multi-parameter analysis is based on the notion of Pareto optimality. Here all relevant parameters are considered in isolation, their values for possible solutions are estimated, and the solutions that behave the best w.r.t. all the parameters at the same time are identified. In other words, solutions that are dominated by some other solution w.r.t. to all of the considered parameters are discarded, and the remaining ones are relevant for the further analysis. This paper is concerned with this second, Pareto-based approach for the multi-parameter analysis.

In order to be able to quantitatively evaluate the security of a system, the expert first needs to formally describe the attack–defense scenarios that the system can be subject to. To perform this task, we employ an extension of attack trees, called *attack–defense trees* (*ADTrees*). The choice of this model has been motivated by its industrial popularity. Attack tree-based analysis is nowadays commonly used by security engineers, thus a solution relying on this formalism will likely be appealing and easy to adopt by security professionals.

An ADTree is a labeled tree representing how an attacker can attack the analyzed system and how a defender can protect the latter against the potential attacks. The scenarios modeled with ADTrees are composed of actions (to be executed by the attacker or a defender) depicted by the labels of the tree nodes. The hierarchical, tree-like structure of an ADTree can be exploited to perform an efficient, quantitative analysis of security scenarios: the actions composing the attacker's and the defender's strategies are decorated with the parameter's values, and the value for the entire scenario is then obtained by propagating these values up to the root node. Unfortunately, the classical bottom-up evaluation assumes that all the nodes in an ADTree represent independent, so in particular different,

actions. In practice, however, the execution of an action may contribute to several attacks. Different nodes of an ADTree may thus hold the same label, and the bottom-up evaluation on such a tree may yield incorrect results. Some alternative methods have been introduced to correctly handle quantitative evaluation on ADTrees with repeated labels, by first translating the tree into another formal object (e.g., a Boolean function or an automaton) and performing the computations on this object. The main drawback of these approaches is that handling these objects can be costly compared to the bottom-up evaluation that is simply achieved with a single-pass traversal of the tree.

Contributions: The aim of this work is to formalize a general framework for efficient, multi-parameter quantitative analysis of security scenarios modeled with ADTrees. Our framework is based on Pareto optimality, and our contributions are the following:

1) We provide a general method for constructing *Pareto attribute domain* – an algebraic structure suitable for multi-parameter optimization of a number of parameters.
2) We prove that Pareto domains enjoy the desirable properties necessary for exploiting two established methods for quantitative evaluation of ADTrees with repeated labels.
3) Our framework allows for simultaneous optimization of a wide class of parameters, including *cost*, *time*, *difficulty* and *probability*, at the same time.
4) We discuss the underlying complexity issues and parameters to be considered when choosing the optimal evaluation method for a given tree.
5) We validate our proposal on a real-world case study borrowed from industry.
6) Finally, we evaluate our framework on numerous trees of various complexity using a proof of concept implementation that we have developed.

Structure: We start by presenting the background knowledge on ADTrees and their quantitative analysis using attributes, in Section II. Section III discusses the problem of attributes' evaluation on ADTrees with repeated labels. Our Pareto-based framework is presented in Section IV, and its empirical validation is performed in Section V. We purposely postpone the description of related work to Section VI, which allows us to compare our proposal with other existing approaches for quantitative, multi-parameter analysis of ADTrees and related models. We conclude our paper in Section VII, where we also identify interesting directions for future work.

II. Background on ADTrees

The objective of this section is to establish notation and introduce the main objects used in this paper, namely ADTrees and attribute domains. In Section II-A, we briefly present ADTrees, set up a term notation for their succinct representation, and define a formal semantics used in this work. We recall the notion of an attribute domain for ADTrees in Section II-B, and illustrate its usage for the classical bottom-up quantitative evaluation of attributes.

A. ADTrees

An *ADTree* [1] is an AND-OR, labeled tree representing a security scenario involving two competing actors – an attacker and a defender. The objective of the attacker is to attack the analyzed system and the objective of the defender is to make the system resistant against the attack. The main goal of one of the actors – called the *proponent* – is stated in the label of the tree's root. This, often high-level and abstract, goal is then recursively refined into subgoals represented by the labels of the remaining nodes. ADTrees considered in this work admit two types of nodes: `OR` and `AND` nodes. The children of an `OR` node represent alternative ways of achieving the node's goal, whereas the children of an `AND` node correspond to the subgoals that all need to be achieved so that the node's goal is achieved. The objective of the other actor – called the *opponent* – is to prevent the proponent from reaching their goal. In order to counter a (sub)goal of the proponent, a node of the opponent is attached to the proponent's node labeled by this (sub)goal. As in the case of the proponent, the goals of the opponent can be refined using the `OR` and `AND` refinements. In addition, the nodes of the opponent can also be countered by the nodes of the proponent. The labels of the nodes that are non-refined represent *basic actions*, i.e., actions that need to be executed by actors to reach their goals.

We use standard graphical notation while drawing ADTrees. The nodes of the attacker are depicted with red ellipses and those of the defender with green rectangles. An arc is used to mark the `AND` nodes. The countermeasures are attached to the nodes they are countering via a dotted line.

In Example 1, we present an ADTree, borrowed from [2], that is used as a running example in this paper. Its graphical representation is given in Figure 1.

Example 1. *In the scenario represented by the ADTree from Figure 1, the proponent is the attacker and the opponent is the defender. The attacker wants to steal money from a victim's account. To achieve this goal, the attacker can use physical means, i.e., learn the victim's PIN, steal their card, and then withdraw cash from an ATM. To learn the PIN, the attacker could force the victim to reveal it or eavesdrop on the victim when they enter the PIN. The victim could prevent the latter by covering the keypad with hand. However, covering the keypad fails if the attacker monitors the keypad with a hidden micro–camera installed at an appropriate spot.*

Instead of attacking from a physical angle, the attacker can steal money by exploiting online banking services. In order to do so, they need to learn the victim's user name and password. Both of these goals can be achieved by creating a fake bank website and using phishing techniques for tricking the account holder into entering their credentials. The attacker could also try to guess what the password and the user name are. Using very strong password would allow the account holder to counter such a guessing attack. Once the attacker obtains the credentials, they use them for logging into the online banking services and execute a transfer. To prevent this type of attack, transfer dispositions might be additionally secured with

two-factor authentication using mobile phone text messages. However, this security measure could be counterattacked by stealing the victim's phone.

ADTrees extend the well-known model of *attack trees* [3], [4] with the countermeasure nodes. The only actor present in attack trees is the attacker who is always the proponent. In attack trees, basic actions correspond to the leaf nodes, which is not necessarily the case in ADTrees. A simple analysis of the tree in Figure 1 shows that `eavesdrop` is a basic action in the scenario from Example 1 but it is not a leaf. In ADTrees, nodes holding basic actions do not have any children belonging to the same actor, i.e., they are non-refined. However, they can still be countered by a child node of the other actor.

As the examples documented in industrial reports [5], [6] show, real-life attack tree-based models often contain several nodes with the same label. When proposing new analysis methods, it is thus important to guarantee that they handle such trees correctly. Following the approach of [2], we assume that the nodes labeled with the same basic action represent exactly the same occurrence of the action. This is for instance the case of the two nodes labeled `phishing`. The idea is that a single phishing attack suffices to get both, the password and the user name of the victim. In the framework developed in this work, two nodes that represent distinct occurrences of an action need to be labeled with (slightly) different labels.

To ease the formal treatment of ADTrees, we employ a term-based notation. The components belonging to the proponent are indexed with p and those belonging to the opponent with o. For $\mathrm{s} \in \{\mathrm{p}, \mathrm{o}\}$, let us denote by $\mathbb{B}^{\mathrm{s}}$ the set of basic actions of the corresponding actor. We assume that the sets $\mathbb{B}^{\mathrm{p}}$ and $\mathbb{B}^{\mathrm{o}}$ are disjoint, and we set $\mathbb{B} := \mathbb{B}^{\mathrm{p}} \cup \mathbb{B}^{\mathrm{o}}$. Formally, ADTrees are terms generated by the grammar (1), where $\mathrm{s} \in \{\mathrm{p}, \mathrm{o}\}$, $\bar{\mathrm{p}} := \mathrm{o}$, $\bar{\mathrm{o}} := \mathrm{p}$ and $\mathtt{b}^{\mathrm{s}} \in \mathbb{B}^{\mathrm{s}}$. The set of all ADTrees is denoted by $\mathbb{T}$.

$$T^{\mathrm{s}} \colon \mathtt{b}^{\mathrm{s}} \mid \mathtt{OR}^{\mathrm{s}}(T^{\mathrm{s}}, \ldots, T^{\mathrm{s}}) \mid \mathtt{AND}^{\mathrm{s}}(T^{\mathrm{s}}, \ldots, T^{\mathrm{s}}) \mid \mathtt{C}^{\mathrm{s}}(T^{\mathrm{s}}, T^{\bar{\mathrm{s}}}) \quad (1)$$

To formally enumerate the situations allowing the proponent to achieve the root goal of an ADTree, formal semantics are used. The semantics employed in this paper represents such situations using pairs $(P, O) \in 2^{\mathbb{B}^{\mathrm{p}}} \times 2^{\mathbb{B}^{\mathrm{o}}}$, where execution by the proponent of all of the actions from P achieves the root goal, provided that none of the actions from O is executed by the opponent. In particular, if $O = \emptyset$, then the proponent executing all of the actions from P achieves the root goal, regardless of the behavior of the opponent. The set semantics was first introduced in [7] and was then used in [2], where quantitative analysis of ADTrees with repeated labels has been studied. We recall the construction of the set semantics in Definition 1. For the sets of pairs of sets of basic actions $X_1, \ldots, X_\ell \subseteq 2^{\mathbb{B}^{\mathrm{p}}} \times 2^{\mathbb{B}^{\mathrm{o}}}$, we set

$$\bigodot_{i=1}^{\ell} X_i := \{(\bigcup_{i=1}^{\ell} P_i, \bigcup_{i=1}^{\ell} O_i) \mid (P_i, O_i) \in X_i\}. \quad (2)$$

Definition 1. The set semantics for ADTrees is a function $\mathcal{S} \colon \mathbb{T} \to 2^{2^{\mathbb{B}^{\mathrm{p}}} \times 2^{\mathbb{B}^{\mathrm{o}}}}$ that assigns to each ADTree a set of pairs of sets of labels, as follows

$$\mathcal{S}\left(\mathtt{b}^{\mathrm{p}}\right) = \left\{\left(\{\mathtt{b}^{\mathrm{p}}\}, \emptyset\right)\right\}, \qquad \mathcal{S}\left(\mathtt{b}^{\mathrm{o}}\right) = \left\{\left(\emptyset, \{\mathtt{b}^{\mathrm{o}}\}\right)\right\},$$

$$\mathcal{S}\left(\mathtt{OR}^{\mathrm{p}}(T_1^{\mathrm{p}}, \ldots, T_\ell^{\mathrm{p}})\right) = \bigcup_{i=1}^{\ell} \mathcal{S}(T_i^{\mathrm{p}}),$$

$$\mathcal{S}\left(\mathtt{AND}^{\mathrm{p}}(T_1^{\mathrm{p}}, \ldots, T_\ell^{\mathrm{p}})\right) = \bigodot_{i=1}^{\ell} \mathcal{S}(T_i^{\mathrm{p}}),$$

$$\mathcal{S}\left(\mathtt{C}^{\mathrm{p}}(T_1^{\mathrm{p}}, T_2^{\mathrm{o}})\right) = \mathcal{S}(T_1^{\mathrm{p}}) \odot \mathcal{S}(T_2^{\mathrm{o}}).$$

$$\mathcal{S}\left(\mathtt{OR}^{\mathrm{o}}(T_1^{\mathrm{o}}, \ldots, T_\ell^{\mathrm{o}})\right) = \bigodot_{i=1}^{\ell} \mathcal{S}(T_i^{\mathrm{o}}),$$

$$\mathcal{S}\left(\mathtt{AND}^{\mathrm{o}}(T_1^{\mathrm{o}}, \ldots, T_\ell^{\mathrm{o}})\right) = \bigcup_{i=1}^{\ell} \mathcal{S}(T_i^{\mathrm{o}}),$$

$$\mathcal{S}\left(\mathtt{C}^{\mathrm{o}}(T_1^{\mathrm{o}}, T_2^{\mathrm{p}})\right) = \mathcal{S}(T_1^{\mathrm{o}}) \cup \mathcal{S}(T_2^{\mathrm{p}}).$$

Given an ADTree T, each element of its set semantics, i.e., a pair $(P, O) \in \mathcal{S}(T)$, is called a *strategy*.

Example 2. *The set semantics of the tree T in Figure 1 is*

$$\begin{aligned}\mathcal{S}(T) = \big\{&(\{\mathtt{force}, \mathtt{card}, \mathtt{cash}\}, \emptyset),\\ &(\{\mathtt{cam}, \mathtt{eav}, \mathtt{card}, \mathtt{cash}\}, \emptyset),\\ &(\{\mathtt{eav}, \mathtt{card}, \mathtt{cash}\}, \{\mathtt{cover}\}),\\ &(\{\mathtt{phish}, \mathtt{log\&trans}\}, \{\mathtt{sms}\}),\\ &(\{\mathtt{phish}, \mathtt{uname}, \mathtt{log\&trans}\}, \{\mathtt{sms}\}),\\ &(\{\mathtt{phish}, \mathtt{pwd}, \mathtt{log\&trans}\}, \{\mathtt{spwd}, \mathtt{sms}\}),\\ &(\{\mathtt{uname}, \mathtt{pwd}, \mathtt{log\&trans}\}, \{\mathtt{spwd}, \mathtt{sms}\}),\\ &(\{\mathtt{phish}, \mathtt{phone}, \mathtt{log\&trans}\}, \emptyset),\\ &(\{\mathtt{phish}, \mathtt{uname}, \mathtt{phone}, \mathtt{log\&trans}\}, \emptyset),\\ &(\{\mathtt{phish}, \mathtt{pwd}, \mathtt{phone}, \mathtt{log\&trans}\}, \{\mathtt{spwd}\}),\\ &(\{\mathtt{uname}, \mathtt{pwd}, \mathtt{phone}, \mathtt{log\&trans}\}, \{\mathtt{spwd}\})\big\}.\end{aligned}$$

The strategy $(\{\mathtt{force}, \mathtt{card}, \mathtt{cash}\}, \emptyset)$ corresponds to the situation where the attacker forces the victim to reveal their PIN, steals their card, and withdraws the cash from an ATM. The second component of this strategy, i.e., the empty set, models that none of the defender's actions is forbidden in order for this scenario to succeed. Indeed, even if the defender executes some (or all) of their three actions, the attacker will still achieve the root goal. In contrast, the strategy $(\{\mathtt{uname}, \mathtt{pwd}, \mathtt{log\&trans}\}, \{\mathtt{spwd}, \mathtt{sms}\})$ depicts a situation where some of the actions cannot be executed by the defender so that the attack succeeds. Here, the attack is composed of guessing the two credentials, logging in, and performing the transfer. This attack can only be successful if the user has not employed a strong password (which would make the guessing impossible) and the SMS-based authentication is not in place. However, if it is not certain whether the SMS-based authentication is implemented or not, the attacker will need to additionally steal the victims phone to be sure that the attack

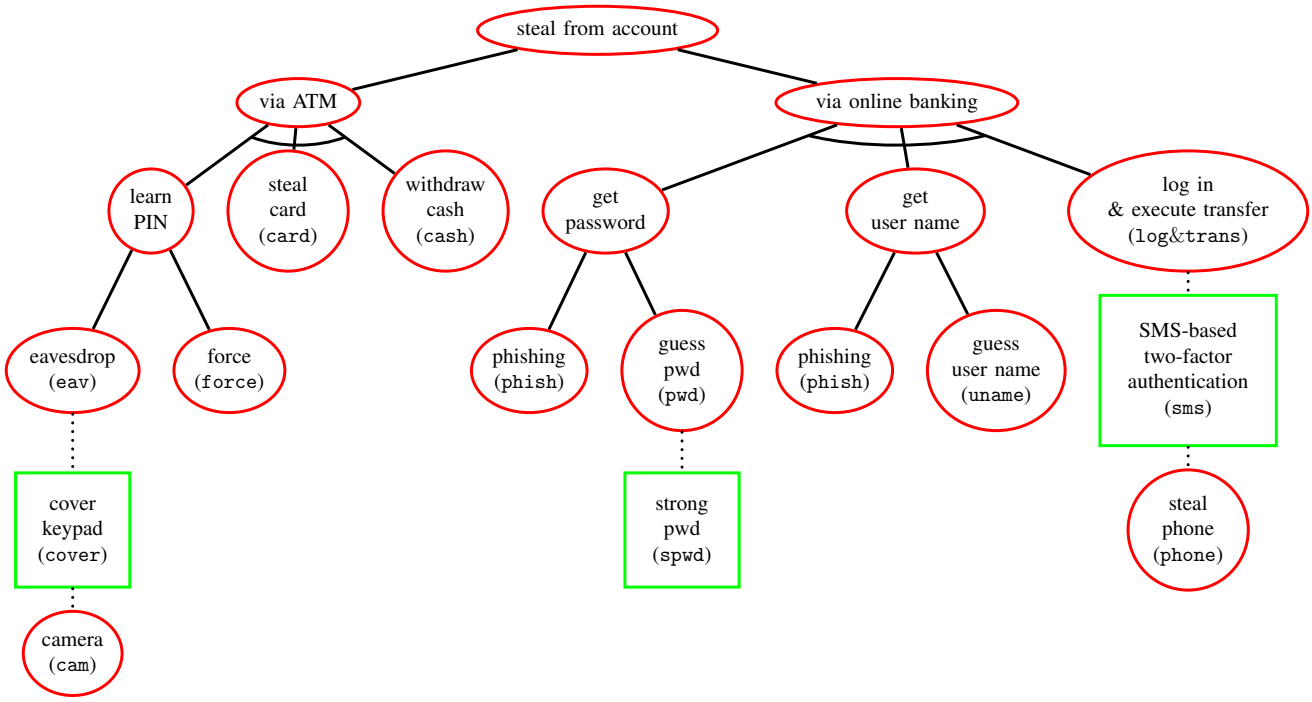

Figure 1: ADTree for stealing money from a bank account.

succeeds. This is for instance the case for the last strategy $(\{\texttt{uname}, \texttt{pwd}, \texttt{phone}, \texttt{log\&trans}\}, \{\texttt{spwd}\})$.

B. Attribute domains for ADTrees

To fully benefit from the process of security modeling using ADTrees, semantic analysis, that, e.g., exhibits possible attacks against a system and highlights its vulnerabilities, should be accompanied by a quantitative analysis of the modeled scenario. A vast number of methods for quantitative analysis of ADTrees exists [2], [8], [9], [10], [11], [12], [13]. Among them, there is a technique based on so called *attributes* [1], [13]. An attribute reflects some quantitative aspect of the modeled scenario, e.g., the *minimal cost* of a successful attack. Classically, to compute the value of an attribute on a given tree, one assigns values to the nodes holding basic actions of the actors (e.g., the minimal cost of their execution), and then propagates these values up to the root of the tree, using appropriate operations at the intermediate nodes. The attributes and the bottom-up procedure described above can be formalized with the help of attribute domains.

Definition 2 (Attribute domain [1])**.** An attribute domain for an attribute α on ADTrees is a tuple

$$A_\alpha = (D_\alpha, \texttt{OR}^{\mathrm{p}}_\alpha, \texttt{AND}^{\mathrm{p}}_\alpha, \texttt{OR}^{\mathrm{o}}_\alpha, \texttt{AND}^{\mathrm{o}}_\alpha, \texttt{C}^{\mathrm{p}}_\alpha, \texttt{C}^{\mathrm{o}}_\alpha),$$

where D_α is a set, and for $\mathrm{s} \in \{\mathrm{p}, \mathrm{o}\}$ and $\texttt{OP} \in \{\texttt{OR}, \texttt{AND}\}$,

1) $\texttt{OP}^{\mathrm{s}}_\alpha$ is an unranked function on D_α,
2) $\texttt{C}^{\mathrm{s}}_\alpha$ is a binary function on D_α.

Examples of some attribute domains for ADTrees are presented in Table I. Their detailed description can be found in [13]. As discussed in [13], attributes are usually defined w.r.t. (at least) one of the actors. For the sake of brevity, while referring to an attribute from Table I, we will omit the actor associated with the attribute. For instance, we will say *minimal cost* instead of *minimal cost for the proponent*.

To formalize the bottom-up evaluation of an attribute α on ADTrees, we use the notion of *basic assignment for* α, being a function $\beta_\alpha \colon \mathbb{B} \to D_\alpha$ assigning values of α to basic actions.

Definition 3 (Bottom–up attribute evaluation [1])**.** Let $(D_\alpha, \texttt{OR}^{\mathrm{p}}_\alpha, \texttt{AND}^{\mathrm{p}}_\alpha, \texttt{OR}^{\mathrm{o}}_\alpha, \texttt{AND}^{\mathrm{o}}_\alpha, \texttt{C}^{\mathrm{p}}_\alpha, \texttt{C}^{\mathrm{o}}_\alpha)$ be an attribute domain, T be an ADTree, and β_α be a basic assignment for α. The value of α for T under β_α obtained via the bottom-up propagation procedure, denoted $\alpha_B(T, \beta_\alpha)$, is defined recursively as

$$\alpha_B(T, \beta_\alpha) := \begin{cases} \beta_\alpha(\mathtt{b}), \text{ if } T = \mathtt{b}, \mathtt{b} \in \mathbb{B}, \\ \texttt{OP}^{\mathrm{s}}_\alpha(\alpha_B(T^{\mathrm{s}}_1, \beta_\alpha), \ldots, \alpha_B(T^{\mathrm{s}}_\ell, \beta_\alpha)), \text{ if } T = \texttt{OP}^{\mathrm{s}}(T^{\mathrm{s}}_1, \ldots, T^{\mathrm{s}}_\ell), \\ \texttt{C}^{\mathrm{s}}_\alpha(\alpha_B(T^{\mathrm{s}}_1, \beta_\alpha), \alpha_B(T^{\bar{\mathrm{s}}}_2, \beta_\alpha)), \text{ if } T = \texttt{C}^{\mathrm{s}}(T^{\mathrm{s}}_1, T^{\bar{\mathrm{s}}}_2), \end{cases}$$

where $\mathrm{s} \in \{\mathrm{p}, \mathrm{o}\}$, and $\texttt{OP} \in \{\texttt{OR}, \texttt{AND}\}$. The index B in the notation $\alpha_B(T, \beta_\alpha)$ refers to the "bottom-up" computation.

We illustrate the bottom-up procedure in Example 3. For a detailed explanation of how to perform the bottom-up evaluation of attributes on ADTrees in practice, we refer the reader to the case studies described in [14], [15].

Example 3. *Consider the tree T from Figure 1 and the attribute domain $A_{\texttt{time}} = (\mathbb{N} \cup \{+\infty\}, \min, +, +, \min, +, \min)$ for the minimal time attribute. Let $\beta_{\texttt{time}}$ be the basic assignment that assigns $+\infty$ to the basic actions of the opponent*

Table I: Selected attribute domains for ADTrees.

Attribute	D_α	$\mathtt{OR}^{\mathrm{p}}_\alpha$	$\mathtt{AND}^{\mathrm{p}}_\alpha$	$\mathtt{OR}^{\mathrm{o}}_\alpha$	$\mathtt{AND}^{\mathrm{o}}_\alpha$	$\mathtt{C}^{\mathrm{p}}_\alpha$	$\mathtt{C}^{\mathrm{o}}_\alpha$
Min. cost for the proponent	$\mathbb{R}_{\geqslant 0} \cup \{+\infty\}$	min	$+$	$+$	min	$+$	min
Max. probability for the proponent	$[0,1]$	max	$\cdot$	$\cdot$	max	$\cdot$	max
Min. time for the proponent	$\mathbb{N} \cup \{+\infty\}$	min	$+$	$+$	min	$+$	min
Max. damage done by the proponent	$\mathbb{R}_{\geqslant 0} \cup \{-\infty\}$	max	$+$	$+$	max	$+$	max
Min. skill level of the proponent	$\mathbb{N} \cup \{+\infty\}$	min	max	max	min	max	min
Proponent's need for special equipment	$\{0,1\}$	min	max	max	min	max	min
Satisfiability for the proponent	$\{\mathrm{false}, \mathrm{true}\}$	$\vee$	$\wedge$	$\wedge$	$\vee$	$\wedge$	$\vee$

and the values given in Table II to those of the proponent. By propagating the values of the basic assignment up to the root, using the operations specified by $A_{\mathtt{time}}$, one obtains

$$\begin{aligned}&\mathtt{time}_B(T, \beta_{\mathtt{time}}) = \\ &\min\Big(\min\big(360 + \min(60, +\infty), 10\big) + 120 + 5, \\ &\min(100, 300 + \infty) + \min(100, 20) + \big(5 + \min(+\infty, 20)\big)\Big) \\ &= \min(135, 145) = 135,\end{aligned}$$

which is the minimal time necessary to achieve the root of T when executing the strategy $(\{\mathtt{force}, \mathtt{card}, \mathtt{cash}\}, \emptyset)$.

Table II: Assignment of *time* and *probability of success* values.

Basic action b	$\beta_{\mathtt{time}}(\mathtt{b})$	$\beta_{\mathtt{prob}}(\mathtt{b})$
cam	60	0.8
eav	360	0.5
force	10	0.3
card	120	0.2
cash	5	0.95
phish	100	0.6
pwd	300	2^{-48}
uname	20	2^{-20}
log&trans	5	0.95
phone	20	0.2

As the bottom-up computation simply propagates the values assigned to the basic actions up to the root of the tree, it involves a number of evaluations of the attribute domain's operations that is linear in the size of the tree. On the downside, it was conceived under the assumption that the basic actions are independent, and therefore might return distorted results if there are repeated basic actions in a tree. This issue was tackled in [2], where alternative methods for the computation of attributes on ADTrees with possibly repeated basic actions were proposed. These methods form the foundation of the framework developed in this work. We summarize them in the next section.

III. Evaluation of attributes in the presence of repeated basic actions

In Section III-A we recall a method of evaluation of attributes that relies on the set semantics and present sufficient conditions for the bottom-up evaluation to return correct results in the presence of repeated basic actions. A method bridging the bottom-up procedure and the evaluation on the set semantics, as well as an important class of attribute domains this method can be applied to, are briefly recalled in Section III-B.

A. Attribute evaluation on the set semantics

Definition 4 (Attribute evaluation on the set semantics [2])**.** Let α be an attribute with the attribute domain $(D_\alpha, \mathtt{OR}^{\mathrm{p}}_\alpha, \mathtt{AND}^{\mathrm{p}}_\alpha, \mathtt{OR}^{\mathrm{o}}_\alpha, \mathtt{AND}^{\mathrm{o}}_\alpha, \mathtt{C}^{\mathrm{p}}_\alpha, \mathtt{C}^{\mathrm{o}}_\alpha)$, T be an ADTree, and β_α a basic assignment. The value of α for T under β_α evaluated on the set semantics, denoted $\alpha_{\mathcal{S}}(T, \beta_\alpha)$, is defined as

$$\begin{aligned}&\alpha_{\mathcal{S}}(T, \beta_\alpha) := \\ &(\mathtt{OR}^{\mathrm{p}}_\alpha)_{(P,O)\in\mathcal{S}(T)}\Big(\mathtt{C}^{\mathrm{p}}_\alpha\big((\mathtt{AND}^{\mathrm{p}}_\alpha)_{\mathtt{b}\in P}\beta_\alpha(\mathtt{b}), (\mathtt{OR}^{\mathrm{o}}_\alpha)_{\mathtt{b}\in O}\beta_\alpha(\mathtt{b})\big)\Big).\end{aligned}$$

In the notation $\alpha_{\mathcal{S}}(T, \beta_\alpha)$, the index $\mathcal{S}$ refers to the computation on the "set semantics".

Example 4. *Let T and $\beta_{\mathtt{time}}$ be as in Example 3. The value of the minimal time of attacking in T evaluated on the set semantics of T (provided in Example 2) is*

$$\begin{aligned}\mathtt{time}_{\mathcal{S}}(T, \beta_{\mathtt{time}}) = &\min(135, 545, +\infty, +\infty, +\infty, +\infty, \\ &+\infty, 125, 145, +\infty, +\infty) = 125.\end{aligned}$$

This value corresponds to the execution of the strategy $(\{\mathtt{phish}, \mathtt{phone}, \mathtt{log\&trans}\}, \emptyset)$.

Observe that the value and the strategy obtained here are different from the ones obtained when using the bottom-up procedure, in Example 3. This is due to the presence of the repeated basic action `phishing` *— each of the two nodes labeled* phish *are treated by the bottom-up procedure as if they were representing two independent phishing attacks. However, in reality, the attacker will get both — user name and password — by launching the same phishing attack.*

While the computation on the set semantics tackles the repeated basic actions properly, its complexity is high, as it requires creation of the set semantics, and the size of the latter might be exponential in the number of nodes in the tree (due to the $\odot$ operation defined by formula (2)). Ideally, one would like to use the fast bottom-up procedure, while being sure that it will return the same result as the computation on the set semantics. The equality of the results of the two methods depends partially on the structure of the attribute domain under consideration, and partially on the presence of repeated basic actions in the tree, as stated in the following result.

Theorem 1 ([2], Theorem 1)**.** *Let T be an ADTree generated by grammar* (1) *and let $A_\alpha = (D_\alpha, \oplus, \otimes, \otimes, \oplus, \otimes, \oplus)$ be an attribute domain, such that the operations $\oplus$ and $\otimes$ are associative and commutative, $\oplus$ is idempotent, and $\otimes$ distributes over $\oplus$. If*

- *there are no repeated labels in T, or*
- *the operator $\otimes$ is idempotent,*

then $\alpha_B(T, \beta_\alpha) = \alpha_{\mathcal{S}}(T, \beta_\alpha)$ holds for any basic assignment β_α.

The attribute domains presented in Table I have the following feature in common. Each of them is of the form $(D_\alpha, \oplus, \otimes, \otimes, \oplus, \otimes, \oplus)$, where $(D_\alpha, \oplus, \otimes)$ is a commutative idempotent semiring. Recall that a commutative idempotent semiring $(R, \oplus, \otimes)$ is a set R equipped with two binary operations that are associative and commutative, where $\oplus$ is idempotent, $\otimes$ distributes over $\oplus$, and the absorbing element of $\otimes$, denoted $\mathtt{a}_\otimes$, is equal to the neutral element $\mathtt{e}_\oplus$ of $\oplus$.

Remark 1. Whereas evaluation of attributes on attack trees is fairly intuitive, the case of ADTrees requires some explanation and care. In the case of attributes defined for the proponent, in order to provide meaningful results, an interpretation of actions of the opponent needs to be fixed, and reflected in the values assigned to them by the basic assignment. In particular, for a number of attribute domains of the form $A_\alpha = (D_\alpha, \oplus, \otimes, \otimes, \oplus, \otimes, \oplus)$, with $(D_\alpha, \oplus, \otimes)$ being a commutative idempotent semiring, under the assumption that a given basic action is executed by the opponent, the value assigned to it is $\mathtt{a}_\otimes$ $(= \mathtt{e}_\oplus)$, whereas the value assigned to the opponent's actions assumed not to be executed is $\mathtt{e}_\otimes$. In consequence, the actions not executed by the opponent do not influence the computation of the attribute, while the executed actions (unless countered by the proponent) absorb the results of the computation corresponding to a given strategy (in the case of the computation on the set semantics) or to a given subtree of the tree (in the case of the bottom-up computation).

Example 5. *An illustration for the above remark has already been provided in Examples 3 and 4, where the value assigned to the basic actions of the opponent was $+\infty$. Observe that $+\infty$ is the absorbing element for the addition and the neutral element for the minimum. Therefore, the result obtained in Example 4 states that if the opponent executes all of their actions, then the minimal time needed for achieving the root goal of T is 125. This is the minimal time needed for the execution of the strategy* $(\{\mathtt{phish}, \mathtt{phone}, \mathtt{log\&trans}\}, \emptyset)$.

B. Evaluation of non–increasing attributes

Among the attribute domains gathered in Table I, the last three are of the form $(D_\alpha, \oplus, \otimes, \otimes, \oplus, \otimes, \oplus)$ satisfying the assumptions of Theorem 1, with the operation $\otimes$ being idempotent. Therefore, the bottom-up procedure can be used to evaluate these attributes on ADTrees containing repeated basic actions. While the remaining attribute domains do not satisfy all of the assumptions of Theorem 1, some of them enjoy a useful property that can be exploited for the purpose of the attribute evaluation on ADTrees with repeated basic actions. This property is captured by the following notion.

Definition 5 (Non-increasing attribute domain [2])**.** An attribute domain A_α is non-increasing if A_α is of the form $(D_\alpha, \oplus, \otimes, \otimes, \oplus, \otimes, \oplus)$, where $(D_\alpha, \oplus, \otimes)$ is a commutative idempotent semiring, such that for every $c, d \in D_\alpha$ the equality $c \oplus (c \otimes d) = c$ holds[1].

We note that from the attribute domains displayed in Table I, only the *maximal damage* domain is not non-increasing. This is because the equality $\max(c, c + d) = c$ does not hold for every $c, d \in \mathbb{R}_{\geqslant 0}$. Note that the *minimal cost*, *minimal time* and the *maximal probability*[2] attribute domains are non-increasing, while they do not satisfy the assumptions of Theorem 1.

An alternative method for computing the value of an attribute, a method that does not require creation of the set semantics of the tree and yet is suitable for non-increasing attribute domains and trees containing repeated basic actions, has been proposed in [2]. The idea behind this method is to temporarily modify the values assigned to the repeated basic actions and perform the bottom-up procedure multiple times. The results obtained in this way are eventually combined in an appropriate manner, yielding the same result as the computation on the set semantics. We refer the reader for details to [2]. The following is implicit in the complexity analysis of this method, as described in [2].

Theorem 2. *Let T be an ADTree generated by grammar* (1)*, with n nodes and k repeated basic actions of the proponent. Let $A_\alpha = (D_\alpha, \oplus, \otimes, \otimes, \oplus, \otimes, \oplus)$ be a non–increasing attribute domain such that a single bottom-up computation $\alpha_B(T, \beta_\alpha)$ is performed in time $\mathcal{O}(f(n))$, for some function $f\colon \mathbb{N} \to \mathbb{R}$. Finally, let $\beta_\alpha\colon \mathbb{B} \to D_\alpha$ be a basic assignment satisfying $\beta_{\alpha|_{\mathbb{B}^o}} \equiv \mathtt{a}_\otimes$. Then, the value of $\alpha_{\mathcal{S}}(T, \beta_\alpha)$ can be computed in time $\mathcal{O}\left(\max(n^2 + f(n), 2^k f(n))\right)$.*

We stress the fact that the complexity of the method from [2] is exponential *in the number of repeated basic actions of the proponent*. Contrarily, the worst-case size of the set semantics is exponential *in the total number of nodes*.

IV. Pareto optimization

We now build upon the methods for the evaluation of attributes on ADTrees with repeated labels recalled in the previous section, and show how they can be exploited in the case of the multi-parameter evaluation. We develop a construction that, given a number of attribute domains, e.g., the domains for the *minimal cost*, the *minimal time*, and the *maximal probability* attributes, results in an attribute domain that can be used for determining strategies that optimize all of these parameters at once. We specify what we mean by "optimal" strategies in Section IV-A. Our construction is presented in Section IV-B and its properties are examined in Section IV-C. Section IV-D tackles the complexity issues related to the application of our framework.

[1]This condition is equivalent to the inequality $d \otimes c \preceq c$, where $\preceq$ stands for the canonical partial order on the semiring D_α, i.e., the order defined by $d \preceq c$ if and only if $d \oplus c = c$. This is the reason for the name *non-increasing*.

[2]The domain $([0, 1], \max, \cdot, \cdot, \max, \cdot, \max)$ is suitable for modeling the *probability of occurrence* as well as the *probability of success* attributes.

A. Pareto points

To compare different strategies while taking multiple attributes related to their execution into account, we assign vectors of values to the strategies. Our main focus is on the attribute domains based on commutative idempotent semirings. Therefore, every set D_i considered in the remainder of this paper is equipped with two binary operations $\oplus_i$ and $\otimes_i$, such that $(D_i, \oplus_i, \otimes_i)$ is a commutative idempotent semiring. Vectors belonging to $D_1 \times \ldots \times D_m$ will be marked in bold, and if $\mathbf{d}$ is a vector, d_i will stand for its ith coordinate. We use $\preceq_i$ to denote the canonical partial order on D_i, defined with $d \preceq_i d'$ iff $d \oplus_i d' = d'$, for $d, d' \in D_i$. Intuitively, $d \preceq_i d'$ iff d' is preferred over d. To compare the elements of the set $D_1 \times \ldots \times D_m$, we use the following standard partial ordering[3] induced by the orders $\preceq_i$.

Definition 6 (Dominance)**.** For $\mathbf{d}, \mathbf{d}' \in D_1 \times \ldots \times D_m$, we say that $\mathbf{d}'$ dominates $\mathbf{d}$ (equivalently, $\mathbf{d}$ is dominated by $\mathbf{d}'$), denoted $\mathbf{d} \preceq \mathbf{d}'$, if the inequality $d_i \preceq_i d'_i$ holds for every $i \in \{1, \ldots, m\}$.

Example 6. *Consider the minimal time and the maximal probability attribute domains (cf. Table I). Observe that the former is based on the commutative idempotent semiring* $(\mathbb{N} \cup \{+\infty\}, \min, +)$*, and the latter on the commutative idempotent semiring* $([0,1], \max, \cdot)$*. To choose strategies optimal w.r.t. both attributes, we consider the set* $(\mathbb{N} \cup \{+\infty\}) \times [0,1]$*. Following Definition 6, a point* (d_1, d_2) *is dominated by a point* (d'_1, d'_2) *if* $\min(d_1, d'_1) = d'_1$ *and* $\max(d_2, d'_2) = d'_2$*. In other words,* $(d_1, d_2) \preceq (d'_1, d'_2)$ *if* $d_1 \geq d'_1$ *and* $d_2 \leq d'_2$*.*

For example, let $D = \{(125, 0.114), (135, 0.057), (145, 2^{-23})\}$ *be the set of points representing the minimal time and the maximal success probability of the strategies* $(\{\texttt{phish}, \texttt{phone}, \texttt{log\&trans}\}, \emptyset)$*,* $(\{\texttt{force}, \texttt{card}, \texttt{cash}\}, \emptyset)$ *and* $(\{\texttt{phish}, \texttt{uname}, \texttt{phone}, \texttt{log\&trans}\}, \emptyset)$*, respectively, under the assignment given by Table II. The points* $(145, 2^{-131})$ *and* $(135, 0.057)$ *are both dominated by* $(125, 0.114)$*.*

If an element of $D_1 \times \ldots \times D_m$ corresponding to the value of a strategy (P, O) is dominated by the value of a strategy (P', O'), e.g., the two strategies are equally likely to succeed, but the cost of execution of (P', O') is smaller, then the proponent has no incentive to execute (P, O). Therefore, the interesting elements of $D_1 \times \ldots \times D_m$ are the ones that are not dominated by others.

Definition 7 (Pareto point)**.** An element $\mathbf{d} \in D \subseteq D_1 \times \ldots \times D_m$ is called a Pareto point of D if it is not dominated by any other element of D, i.e., if $\mathbf{d} \not\preceq \mathbf{d}'$ holds for every $\mathbf{d}' \in D, \mathbf{d}' \neq \mathbf{d}$.

It is known that if a partially ordered set is finite, then the set of all of its Pareto points is unique (Corollary 1 in [16]).

[3]In the general case of partially ordered sets (not necessarily commutative idempotent semirings) the definitions are analogous, cf. [16].

Definition 8 (Pareto frontier)**.** The set of all Pareto points of a finite set $D \subseteq D_1 \times \ldots \times D_m$, denoted $\max(D)$[4], is called Pareto frontier of D.

Example 7. *Consider again the two domains and the set* D *from Example 6. As already observed, the point* $(125, 0.114)$ *dominates the remaining points of* D*. Thus, we have*

$$\max(D) = \{(125, 0.114)\}.$$

Our ultimate goal is to identify values of strategies that are not dominated by values corresponding to the execution of other strategies. In other words, the final result of our analysis will be a set whose every element is a Pareto point.

Definition 9 (Pareto optimal set)**.** A finite set $D \subseteq D_1 \times \ldots \times D_m$ satisfying $D = \max(D)$ is called a Pareto optimal set. We use $P(D_1 \times \ldots \times D_m)$ to denote the set of all Pareto optimal sets in $D_1 \times \ldots \times D_m$.

The considerations in Example 6 and 7 show that D defined in Example 6 is not a Pareto optimal set.

B. Pareto attribute domains

We are now ready to develop a general method for combining attribute domains into a single domain suitable for determining Pareto optimal strategies in ADTrees.

For $i \in \{1, \ldots, m\}$, let A_{α_i} be the attribute domain $(D_i, \oplus_i, \otimes_i, \otimes_i, \oplus_i, \otimes_i, \oplus_i)$. Given basic assignments β_{α_i} for the attributes α_i, we create a new assignment, which assigns the singleton $\{(\beta_{\alpha_1}(\mathtt{b}), \ldots, \beta_{\alpha_m}(\mathtt{b}))\}$ to each basic action $\mathtt{b} \in \mathbb{B}$. Note that this singleton is a Pareto optimal set, and it contains the optimal value corresponding to the execution of $\mathtt{b}$. Such singletons will be combined using appropriate operations, eventually resulting in a Pareto optimal set of values corresponding to strategies in an ADTree. We now define these operations.

For $\mathbf{d}, \mathbf{d}' \in D_1 \times \ldots \times D_m$, let

$$\mathbf{d} \otimes \mathbf{d}' := (d_1 \otimes_1 d'_1, \ldots, d_m \otimes_m d'_m), \tag{3}$$

and, with a slight abuse of notation, let

$$D \otimes D' := \{\mathbf{d} \otimes \mathbf{d}' \colon \mathbf{d} \in D, \mathbf{d}' \in D'\}, \tag{4}$$

$$D \mathbin{\widehat{\otimes}} D' := \max(D \otimes D'), \tag{5}$$

$$D \mathbin{\widehat{\oplus}} D' := \max(D \cup D'), \tag{6}$$

for $D, D' \in P(D_1 \times \ldots \times D_m)$.

Example 8. *Consider the minimal time and the maximal probability attribute domains, cf. Example 6. The operation defined by formula* (3) *will in this case be* $\mathbf{d} \otimes \mathbf{d}' = (d_1 + d'_1, d_2 \cdot d'_2)$*, for* $\mathbf{d}, \mathbf{d}' \in (\mathbb{N} \cup \{+\infty\}) \times [0,1]$*.*

The intuition behind the above construction is the following. Suppose that two sets D and D' contain Pareto optimal values corresponding to the achievement of two different subgoals by the proponent in a tree with no repeated basic actions. If in

[4]The choice of the $\max(\cdot)$ notation is dictated by the fact that Pareto points are the maximal elements w.r.t. the dominance relation.

order to achieve the root goal of T the proponent has to achieve at least one of the two subgoals, then the set of Pareto optimal values of achieving the root goal is computed as $D \mathbin{\widehat{\oplus}} D'$: this operation first gathers all the values corresponding to the strategies achieving the root goal in a single set, and then returns the Pareto frontier of this set. Similarly, if the proponent had to achieve both of the aforementioned goals, then the Pareto optimal values of strategies in T would be obtained by computing $D \mathbin{\widehat{\otimes}} D'$: here the result is the Pareto frontier of the set of all possible values corresponding to simultaneous achievement of the two subgoals.

Given the above construction, the values of Pareto optimal strategies can be obtained using the attribute domain $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$. Throughout the rest of the paper, we refer to the attribute domains resulting from the above construction as *Pareto attribute domains*.

Definition 10 (Pareto attribute domain)**.** A *Pareto attribute domain* is an algebraic structure of the form $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$, for some attribute domains $A_{\alpha_i} = (D_i, \oplus_i, \otimes_i, \otimes_i, \oplus_i, \otimes_i, \oplus_i)$, for $i \in \{1, \ldots, m\}$, with $(D_i, \oplus_i, \otimes_i)$ being commutative idempotent semirings, and with $\widehat{\oplus}$ and $\widehat{\otimes}$ defined by formulas (3)–(6). We say that the Pareto attribute domain $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$ is *induced by* the attribute domains A_{α_i}, for $i \in \{1, \ldots, m\}$.

Theorem 3 presents our main results about the Pareto attribute domains.

Theorem 3. *A Pareto attribute domain* $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$ *induced by the attribute domains* A_{α_i}, $i \in \{1, \ldots, m\}$, *is an attribute domain (in the sense of Definition 2), and* $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes})$ *is a commutative idempotent semiring.*

Furthermore, if the domains A_{α_i}, $i \in \{1, \ldots, m\}$, *are non-increasing, then the induced Pareto attribute domain is also non-increasing.*

Before presenting its proof, we briefly discuss the immediate consequences of Theorem 3. The first of them follows from Theorem 1: if there are no repeated basic actions in an ADTree, then the evaluation of a number of attributes having domains based on commutative idempotent semirings can be performed using a single bottom-up procedure. Second, if a tree contains repeated basic actions and the Pareto attribute domain is induced by non-increasing attribute domains, then, by Theorem 2, the algorithm presented in [2] can be applied, and the values of optimal strategies can still be obtained without the need of constructing the set semantics of the entire tree. Third, note that if a Pareto domain is induced by attribute domains whose multiplicative operations are idempotent, the operation $\widehat{\otimes}$ is itself idempotent. Therefore, again due to Theorem 1, in such a case the evaluation of a Pareto attribute can be performed using a single bottom-up procedure.

The above discussion is summarized in the following theorem, which captures the main contributions of our work.

Theorem 4. *Let* T *be an ADTree generated by grammar* (1) *and let* $A_{Par} = (P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$ *be a Pareto attribute domain induced by the attribute domains* A_{α_i}, $i \in \{1, \ldots, m\}$. *Then*

- *if there are no repeated labels in* T*, then the equality* $Par_B(T, \beta_{Par}) = Par_{\mathcal{S}}(T, \beta_{Par})$ *holds for any basic assignment* β_{Par}*,*
- *if the operator* $\otimes_i$ *is idempotent, for every* $i \in \{1, \ldots, m\}$*, then the equality* $Par_B(T, \beta_{Par}) = Par_{\mathcal{S}}(T, \beta_{Par})$ *holds for any basic assignment* β_{Par}*,*
- *if* A_{α_i} *is a non-increasing attribute domain, for every* $i \in \{1, \ldots, m\}$*, and* β_{Par} *is a basic assignment satisfying* $\beta_{Par|_{\mathbb{B}^o}} \equiv \mathtt{a}_{\widehat{\otimes}}$*, then the value of* $Par_{\mathcal{S}}(T, \beta_{Par})$ *can be computed using the method of [2].*

C. Proof of Theorem 3

Our proof of Theorem 3 exploits some basic properties of the dominance relation and of the Pareto frontier, stated in Lemma 1–4. Recall that, for every $i \in \{1, \ldots, m\}$, $(D_i, \oplus_i, \otimes_i)$ is an idempotent commutative semiring and that the dominance relation $\preceq$ in $D_1 \times \ldots \times D_m$ is defined w.r.t. canonical partial orders $\preceq_i$.

Lemma 1. *Let* $\mathbf{d}, \mathbf{d}'$ *and* $\mathbf{d}''$ *be elements of* $D_1 \times \ldots \times D_m$*. Then,*

1) *if* $\mathbf{d}' \preceq \mathbf{d}''$*, then* $\mathbf{d} \otimes \mathbf{d}' \preceq \mathbf{d} \otimes \mathbf{d}''$*,*
2) *if the relation* $d \otimes_i d' \preceq_i d'$ *holds for every* $i \in \{1, \ldots, m\}$ *and for every* $d, d' \in D_i$*, then* $\mathbf{d} \otimes \mathbf{d}' \preceq \mathbf{d}'$*.*

Proof: Since for every $i \in \{1, \ldots, m\}$, $(D_i, \oplus_i, \otimes_i)$ is an idempotent commutative semiring, therefore, for $d, d', d'' \in D_i$ we have that if $d' \preceq_i d''$, then

$$d \otimes_i d'' = d \otimes_i (d' \oplus_i d'') = (d \otimes_i d') \oplus_i (d \otimes_i d''),$$

meaning that $d \otimes_i d' \preceq_i d \otimes_i d''$. Together with definition of the dominance relation, this implies the first statement.

The second statement follows immediately from the definition of $\mathbf{d} \otimes \mathbf{d}'$, defined by (3) on page 7, and the definition of the dominance relation. ∎

Lemma 2. *If* A *and* B *are finite subsets of* $D_1 \times \ldots \times D_m$*, then*

1) $\max(A \cup B) \subseteq \max(A) \cup \max(B)$*,*
2) $\max(A \otimes B) \subseteq \max(A) \otimes \max(B)$*.*

Proof: For a proof of the first of the two statements, let $\mathbf{d} \in \max(A \cup B)$. Since $\mathbf{d}$ is not dominated by any other element of $A \cup B$, it follows that if $\mathbf{d} \in A$, then $\mathbf{d} \in \max(A)$, and if $\mathbf{d} \in B$, then $\mathbf{d} \in \max(B)$. Hence, $\mathbf{d} \in \max(A) \cup \max(B)$.

Now, let $\mathbf{d} = \mathbf{d}_A \otimes \mathbf{d}_B \in \max(A \otimes B)$ for some $\mathbf{d}_A \in A$ and $\mathbf{d}_B \in B$. Towards a contradiction, suppose that $\mathbf{d} \notin \max(A) \otimes \max(B)$. Then, there exist elements $\mathbf{d}'_A \in \max(A)$, $\mathbf{d}'_B \in \max(B)$, such that $\mathbf{d}'_A$ dominates $\mathbf{d}_A$ and $\mathbf{d}'_B$ dominates $\mathbf{d}_B$, with $\mathbf{d}'_A \neq \mathbf{d}_A$ or $\mathbf{d}'_B \neq \mathbf{d}_B$. Since $\mathbf{d} \notin \max(A) \otimes \max(B)$, it follows that $\mathbf{d} \neq \mathbf{d}'_A \otimes \mathbf{d}'_B$. Furthermore, by Lemma 1, it holds that $\mathbf{d} \preceq \mathbf{d}'_A \otimes \mathbf{d}'_B$. This contradicts the choice of $\mathbf{d}$ as a Pareto point in $A \otimes B$. ∎

Lemma 3. *If A and B are finite subsets of $D_1 \times \ldots \times D_m$, then* $\max(\max(A) \cup B) = \max(A \cup B)$.

Proof: Let $\mathbf{d} \in \max(A \cup B)$. Observe that $\mathbf{d} \in \max(A) \cup B$, by Lemma 2. Furthermore, since $\mathbf{d}$ is not dominated by any of the points in $A \cup B$, it is also not dominated by any of the points in $\max(A) \cup B$. This proves that $\max(\max(A) \cup B) \supseteq \max(A \cup B)$.

For a proof of the inclusion $\max(\max(A) \cup B) \subseteq \max(A \cup B)$, let $\mathbf{d}$ be a Pareto point in $\max(A) \cup B$. Suppose that $\mathbf{d}$ is not a Pareto point in $A \cup B$. Then there exists $\mathbf{d}' \in A \cup B$, $\mathbf{d}' \neq \mathbf{d}$, such that $\mathbf{d} \preceq \mathbf{d}'$. Since $\mathbf{d}$ is not dominated by any element of B, it follows that $\mathbf{d}' \in A$. But then, since $\preceq$ is a transitive relation, every $\mathbf{d}'' \in \max(A)$ that dominates $\mathbf{d}'$ dominates also $\mathbf{d}$. This contradicts the choice of $\mathbf{d}$. ∎

Lemma 4. *If A and B are finite subsets of $D_1 \times \ldots \times D_m$, then* $\max(\max(A) \otimes B) = \max(A \otimes B)$.

Proof: For a proof of the inclusion $\max(\max(A) \otimes B) \subseteq \max(A \otimes B)$, let $\mathbf{d} \in \max(\max(A) \otimes B)$. Towards a contradiction, suppose that $\mathbf{d}$ is not a Pareto point in $A \otimes B$. This implies that there exist elements $\mathbf{d}_A \in A$ and $\mathbf{d}_B \in B$ such that $\mathbf{d} \preceq \mathbf{d}_A \otimes \mathbf{d}_B$ and $\mathbf{d} \neq \mathbf{d}_A \otimes \mathbf{d}_B$. Let $\mathbf{d}'_A \in \max(A)$ be such that $\mathbf{d}_A \preceq \mathbf{d}'_A$. Then

$$\mathbf{d} \preceq \mathbf{d}_A \otimes \mathbf{d}_B \preceq \mathbf{d}'_A \otimes \mathbf{d}_B,$$

by Lemma 1. Since $\mathbf{d}'_A \otimes \mathbf{d}_B \in \max(A) \otimes B$, this contradicts the choice of $\mathbf{d}$.

Assume now that $\mathbf{d}$ is a Pareto point in $A \otimes B$. Observe that $\mathbf{d} \in \max(A) \otimes B$, by Lemma 2. Since $\mathbf{d}$ is not dominated by any element of $A \otimes B$, it is in particular not dominated by any element of $\max(A) \otimes B$. Therefore, $\mathbf{d}$ is a Pareto point in $\max(A) \otimes B$. ∎

We are now ready to prove Theorem 3.

Proof. We begin with proving that $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes})$ is a commutative idempotent semiring. Since a binary associative operation can be modeled with an unranked operator, this immediately implies that $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$ is an attribute domain.

For $A \in P(D_1 \times \ldots \times D_m)$, we have

$$A \,\widehat{\oplus}\, A = \max(A \cup A) = \max(A) = A,$$

i.e., the operation $\widehat{\oplus}$ is idempotent. It is easy to verify that both $\widehat{\oplus}$ and $\widehat{\otimes}$ are commutative and that $\mathtt{a}_{\widehat{\otimes}} = \{(\mathtt{a}_{\otimes_1}, \ldots, \mathtt{a}_{\otimes_m})\}$. Since $\mathtt{a}_{\otimes_i} = \mathtt{e}_{\oplus_i}$ for every $i \in \{1, \ldots, m\}$, together with the definitions of canonical partial orders and Definition 6 this implies that $\mathtt{a}_{\widehat{\otimes}}$ is dominated by every other element of $D_1 \times \ldots \times D_m$. Therefore, for any $D \in P(D_1 \times \ldots \times D_m)$, we have that $D \,\widehat{\oplus}\, \mathtt{a}_{\widehat{\otimes}} = \max(D \cup \mathtt{a}_{\widehat{\otimes}}) = \max(D) = D$. This proves that $\mathtt{e}_{\widehat{\oplus}} = \mathtt{a}_{\widehat{\otimes}}$.

The associativity of the two operations follows from Lemma 3 and 4. Namely, we have

$$\begin{aligned}(A \,\widehat{\oplus}\, B) \,\widehat{\oplus}\, C &= \max(\max(A \cup B) \cup C) = \max(A \cup B \cup C) \\ &= \max(A \cup \max(B \cup C)) = A \,\widehat{\oplus}(B \,\widehat{\oplus}\, C)\end{aligned}$$

and

$$\begin{aligned}(A \,\widehat{\otimes}\, B) \,\widehat{\otimes}\, C &= \max(\max(A \otimes B) \otimes C) = \max(A \otimes B \otimes C) \\ &= \max(A \otimes \max(B \otimes C)) = A \,\widehat{\otimes}(B \,\widehat{\otimes}\, C).\end{aligned}$$

We prove that $\widehat{\otimes}$ distributes over $\widehat{\oplus}$ in a similar way

$$\begin{aligned}&A \,\widehat{\otimes}(B \,\widehat{\oplus}\, C) = \max(A \otimes \max(B \cup C)) \overset{\text{Lemma 4}}{=} \\ &\max(A \otimes (B \cup C)) = \max(A \otimes B \cup A \otimes C) \overset{\text{Lemma 3}}{=} \\ &\max(\max(A \otimes B) \cup \max(A \otimes C)) = (A \,\widehat{\otimes}\, B) \,\widehat{\oplus}(A \,\widehat{\otimes}\, C).\end{aligned}$$

The above reasoning proves that $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes})$ is a commutative idempotent semiring and that $(P(D_1 \times \ldots \times D_m), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus})$ is an attribute domain.

Assume now that the domains $(D_i, \oplus_i, \otimes_i, \otimes_i, \oplus_i, \otimes_i, \oplus_i)$ are non-increasing, for $i \in \{1, \ldots, m\}$. To prove the second statement of the theorem, it remains to prove that for every $A, B \in P(D_1 \times \ldots \times D_m)$ the equality $A \,\widehat{\oplus}(A \,\widehat{\otimes}\, B) = A$ holds. Let $A, B \in P(D_1 \times \ldots \times D_m)$. Observe that, since the domains $(D_i, \oplus_i, \otimes_i, \otimes_i, \oplus_i, \otimes_i, \oplus_i)$ are non-increasing, the second item of Lemma 1 implies that $\max(A \cup (A \otimes B)) = \max(A)$. Furthermore, since A is a Pareto optimal set, the equality $\max(A) = A$ holds. Therefore, we have

$$\begin{aligned}A \,\widehat{\oplus}(A \,\widehat{\otimes}\, B) &= \max(A \cup \max(A \otimes B)) \overset{\text{Lemma 3}}{=} \\ &\max(A \cup (A \otimes B)) = \max(A) = A.\end{aligned}$$

The proof of Theorem 3 is complete. □

D. Complexity issues

Theorems 1–3, summarized in Theorem 4, provide a general framework for a convenient multi-objective analysis of scenarios modeled with ADTrees. Before illustrating its applicability, we discuss its complexity.

We note that even in the simplest case of attack trees with a single *minimal cost* attribute domain, the problem of determining a cheapest strategy is known to be NP-hard [17]. Theorems 1 and 2 indicate that this difficulty originates from the presence of repeated basic actions of the proponent. One could therefore hope for the multi-objective optimization to also be easier in trees with no repeated basic actions. Unfortunately, this is not necessarily the case, due to the number of possible Pareto optimal strategies having unique values. The following construction illustrates this issue.

Example 9. *Let $m \geq 2$ be an even integer and let $T = \mathtt{AND}^{\mathrm{P}}(\mathtt{OR}^{\mathrm{P}}(\mathtt{b}_1, \mathtt{b}_2), \mathtt{OR}^{\mathrm{P}}(\mathtt{b}_3, \mathtt{b}_4), \ldots, \mathtt{OR}^{\mathrm{P}}(\mathtt{b}_{m-1}, \mathtt{b}_m))$. Consider a Pareto domain induced by m minimal cost attribute domains and a basic assignment that assigns to the action $\mathtt{b}_i$ a vector admitting 1 on the ith coordinate and 0 on each of the remaining $m-1$ coordinates. Then, every pair of the form $(\{\mathtt{b}_{i_1}, \mathtt{b}_{i_2}, \ldots, \mathtt{b}_{i_{m/2}}\}, \emptyset)$, where $i_j \in \{2j-1, 2j\}$, is a Pareto optimal strategy in T, and the value corresponding to such a strategy is unique. The number of such strategies is $2^{m/2}$.*

If the number of domains inducing a Pareto domain is small, then the time and space complexities of the methods of attribute evaluation depend mostly on two factors: the size

of the set semantics and the number k of repeated basic actions of the proponent in the considered tree. In the case when k is big and the number of strategies is small, it is better to use the computation on the set semantics. This is obviously due to the fact that the time complexity of the method of [2] is exponential in k, cf. Theorem 2. If k is small and the number of strategies in the tree is big, then the method of [2] will perform better. This intuition is supported by the experimental results presented in Section V-B. These results provide also some indications towards making the meaning of the words "big" and "small" more precise for particular use cases.

In order to choose the method of attribute evaluation that would perform better for a given tree, one should therefore count the repeated basic actions of the proponent, and estimate the number of strategies, without creating the set semantics itself. The former is a straightforward task that can be performed quickly. To achieve the latter, a single bottom-up computation of a specific attribute suffices, as described in the following lemma.

Lemma 5. *Let* `SetSemBound` *be an attribute with the attribute domain* $A_{\texttt{SetSemBound}} = (\mathbb{N}, +, \cdot, \cdot, +, \cdot, +)$*, where* $\cdot$ *is the multiplication operator. Let* $\beta_{\texttt{SetSemBound}} \equiv 1$ *be a basic assignment of* `SetSemBound`*. Then the inequality*

$$|\,\mathcal{S}(T)| \leq \texttt{SetSemBound}_B(T, \beta_{\texttt{SetSemBound}}) \tag{7}$$

holds for every ADTree T *generated by grammar* (1)*.*

Proof: The proof is based on the proof of Theorem 1 from [2]. It relies on the fact that the set semantics can be seen as an attribute for ADTrees, having the attribute domain $A_{\mathcal{S}} = (D_{\mathcal{S}}, \cup, \odot, \odot, \cup, \odot, \cup)$, where $D_{\mathcal{S}} = 2^{2^{\mathbb{B}^{\mathrm{P}}} \times 2^{\mathbb{B}^{\mathrm{O}}}}$, and where $\odot$ is defined by formula (2). For the basic assignment

$$\beta_{\mathcal{S}}(\mathtt{b}) = \begin{cases} \{(\{\mathtt{b}\}, \emptyset)\} \text{ if } \mathtt{b} \in \mathbb{B}^{\mathrm{P}}, \\ \{(\emptyset, \{\mathtt{b}\})\} \text{ otherwise,} \end{cases}$$

it holds that $\mathcal{S}(T) = \mathcal{S}_B(T, \beta_{\mathcal{S}})$, by Definition 1. Note that the algebraic structure $(D_{\mathcal{S}}, \cup, \odot)$ constitutes a commutative idempotent semiring. Therefore, the operation $\odot$ distributes over $\cup$, and so the result of the bottom-up procedure $\mathcal{S}_B(T, \beta_{\mathcal{S}})$ can be represented as

$$\begin{aligned} \mathcal{S}_B(T, \beta_{\mathcal{S}}) = {} & (\beta_{\mathcal{S}}(\mathtt{b}^1_1) \odot \beta_{\mathcal{S}}(\mathtt{b}^1_2) \odot \ldots \odot \beta_{\mathcal{S}}(\mathtt{b}^1_{k_1})) \cup \\ & \ldots \\ & \cup (\beta_{\mathcal{S}}(\mathtt{b}^i_1) \odot \beta_{\mathcal{S}}(\mathtt{b}^i_2) \odot \ldots \odot \beta_{\mathcal{S}}(\mathtt{b}^i_{k_i})) \cup \\ & \ldots \\ & \cup (\beta_{\mathcal{S}}(\mathtt{b}^l_1) \odot \beta_{\mathcal{S}}(\mathtt{b}^l_2) \odot \ldots \odot \beta_{\mathcal{S}}(\mathtt{b}^l_{k_l})). \end{aligned} \tag{8}$$

Similarly, since $(\mathbb{N}, +, \cdot)$ is a commutative semiring, the result of the bottom-up computation of the `SetSemBound` attribute can be represented as

$$\begin{aligned} \texttt{SetSemBound}_B(T, \beta_{\texttt{SetSemBound}}) = {} & (1 \cdot 1 \cdot \ldots \cdot 1) + \\ & \ldots \\ & + (1 \cdot 1 \cdot \ldots \cdot 1) + \\ & \ldots \\ & + (1 \cdot 1 \cdot \ldots \cdot 1). \end{aligned} \tag{9}$$

By applying the distributivity rule to the expression $\texttt{SetSemBound}_B(T, \beta_{\texttt{SetSemBound}})$ in the same order as it was applied to $\mathcal{S}_B(T, \beta_{\mathcal{S}})$, one obtains representation (9) whose ith term corresponds to the ith term of (8), i.e., the ith term of (9) is a product of k_i ones.

From definitions of the basic assignment $\beta_{\mathcal{S}}$ and the operation $\odot$ it follows that, for every $i \in \{1, \ldots, n\}$, the ith term

$$\beta_{\mathcal{S}}(\mathtt{b}^i_1) \odot \beta_{\mathcal{S}}(\mathtt{b}^i_2) \odot \cdots \odot \beta_{\mathcal{S}}(\mathtt{b}^i_{k_i})$$

of representation (8) is a set consisting of exactly one pair of sets. Let us denote this term with $\{(P_i, O_i)\}$. Observe that since $\mathcal{S}(T) = \mathcal{S}_B(T, \beta_{\mathcal{S}})$, we have $(P_i, O_i) \in \mathcal{S}(T)$ for every i, and, conversely, for every $(P, O) \in \mathcal{S}(T)$ there exists at least one i such that $(P, O) = (P_i, O_i)$. Therefore, the number of terms of (8) is at least $|\,\mathcal{S}(T)|$. Since this number is equal to $\texttt{SetSemBound}_B(T, \beta_{\texttt{SetSemBound}})$, by (9), the claim follows. ■

Lemma 5 shows that an upper bound on the number of strategies can be found in time linear in the size of the tree. We note that the bound provided by the inequality (7) is tight, as it can be observed in Table III. Nevertheless, the difference between the bound and the actual size of the set semantics can be arbitrarily large. For instance, there is one element in the set semantics of the tree $\texttt{AND}^{\mathrm{P}}(\texttt{OR}^{\mathrm{P}}(\mathtt{b}, \mathtt{b}), \ldots, \texttt{OR}^{\mathrm{P}}(\mathtt{b}, \mathtt{b}))$, while the bound is equal to 2 to the power equal to the number of `OR` nodes.

From the practical point of view, having an easily computable formula for a non-trivial *lower bound* on the size of the set semantics would be more useful. We finish this section by noting that such a lower bound cannot be computed using a bottom-up procedure that would simply propagate natural numbers throughout the tree. This is the case because such a procedure would have to yield 1 for every attack tree in which all the leaf nodes bear the same label, irrespective of the tree structure. To obtain a non-trivial lower bound one would have to propagate, along a number, some additional information about the repeated basic actions seen so far in the tree.

V. Empirical validation

In order to validate our approach, we have created a Python package [18] for manipulating ADTrees. Among other functionalities, the package supports the usage of Pareto domains. In Section V-A, we show the practicality of our approach on a real-world case study. Experimental results illustrating the approach's scalability and the differences between the two methods for attribute evaluation are presented in Section V-B.

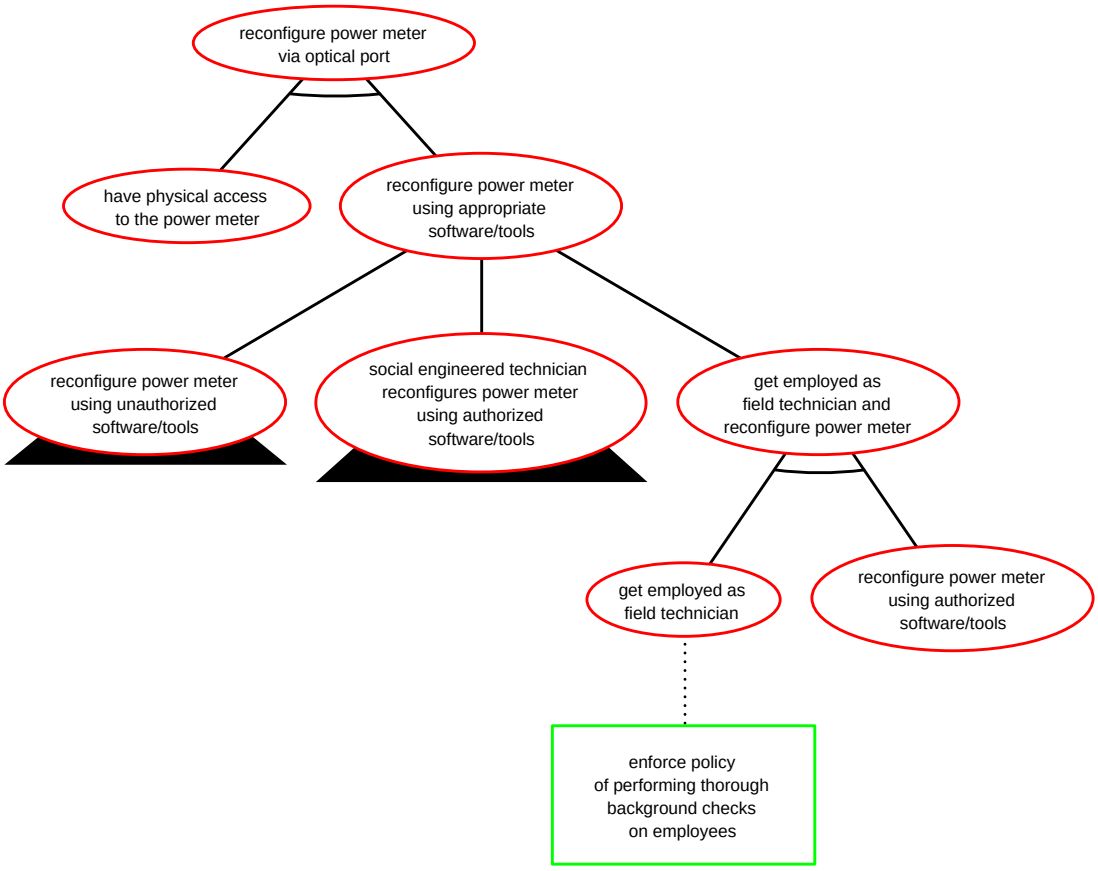

Figure 2: The top part of ADTree for reconfiguring a power meter.

A. Case study

Electricity theft is a serious issue [19], [20] that generates huge financial losses yearly across the world [21]. In this study, we consider the problem, analyzed by the U.S. Department of Energy in [5], where a customer (an attacker) reconfigures their power meter in order to lower the recorded electricity consumption of their household. A detailed description and further analysis of the considered scenario can be found in [15]. Here we focus on multi-parameter evaluation using Pareto domains.

The starting point for our analysis was an attack tree described in Section 2.3 of [5]. We complemented it with additional attack vectors and possible countermeasures, based on [22], [23], [24]. The resulting ADTree contains 68 nodes and 5 repeated basic actions. Its main characteristics relevant for this study are gathered in Table III, where n is the total number of nodes and k is the number of repeated basic actions of the proponent. The top part of the tree is illustrated in Figure 2 and the entire model is accessible at https://github.com/wwidel/pareto-case-study.

Table III: Parameters of ADTree considered in the case study.

n	k	$\lvert\mathcal{S}(T)\rvert$	$\lvert\mathcal{S}(T)\rvert$ bound of Lemma 5
68	5	33	33

Our goal is to analyze the impact of the deployment of different countermeasures mitigating the problem of electricity theft. To perform such a "what-if" analysis, we consider five scenarios ($S_1 - S_5$) listed in Table IV, where different countermeasures have been implemented. Each of the five scenarios is analyzed using the Pareto attribute domain induced by the domains for *minimal cost*, *minimal cybersecurity skill level*, *minimal technical skill level*, *minimal social skill level*, and *minimal time* attributes. The values they can attain and the meaning behind them are described in Table V. The induced Pareto domain is $\big(P(\mathcal{D}), \widehat{\oplus}, \widehat{\otimes}, \widehat{\otimes}, \widehat{\oplus}, \widehat{\otimes}, \widehat{\oplus}\big)$, where

$$\mathcal{D} = \big(\mathbb{R}_{\geqslant 0} \cup \{+\infty\}\big) \times \{0,1,2,3,+\infty\}^3 \times \{0,10,10^2,10^3,10^4,+\infty\},$$

and operations $\widehat{\otimes}$, $\widehat{\oplus}$ are given by equations (4)–(6), where $\otimes$ is defined for $\mathbf{d}, \mathbf{d}' \in \mathcal{D}$ by

$$\mathbf{d} \otimes \mathbf{d}' := (d_1 + d_1', \max(d_2, d_2'), \ldots, \max(d_5, d_5')).$$

Values for basic actions of the attacker, gathered in Table VI, were assigned jointly by seven participants presenting basic knowledge of the power meter technology and having access to relevant sources [5], [22], [23], [24]. A detailed description

Table IV: The Pareto optimal values in scenarios S_1, S_2, S_3, S_4, and S_5 (see Table VII for the meaning of $\mathtt{d}_4, \mathtt{d}_5, \mathtt{d}_7$).

Scenario	S_1	S_2	S_3	S_4	S_5
Countermeasures implemented	none	$\mathtt{d}_7$	$\mathtt{d}_4$	$\mathtt{d}_4, \mathtt{d}_7$	$\mathtt{d}_4, \mathtt{d}_5, \mathtt{d}_7$
Pareto optimal values	$(14, 1, 2, 0, 10^2)$ $(0, 0, 1, 3, 10^2)$ $(0, 0, 0, 3, 10^3)$ $(0, 0, 2, 1, 10^3)$	$(14, 1, 2, 0, 10^2)$ $(0, 0, 1, 3, 10^2)$ $(0, 0, 0, 3, 10^3)$	$(14, 1, 2, 0, 10^2)$ $(0, 0, 1, 3, 10^2)$ $(0, 0, 0, 3, 10^3)$ $(0, 0, 2, 1, 10^3)$	$(14, 1, 2, 0, 10^2)$ $(0, 0, 1, 3, 10^2)$ $(0, 0, 0, 3, 10^3)$	$(14, 1, 2, 0, 10^2)$ $(0, 0, 1, 3, 10^2)$ $(0, 0, 0, 3, 10^3)$

Table V: Attributes for the case study of power meter tampering

Attribute	**Values**	**Attribute domain**
Minimal cost	Non-negative real values (in euros)	$(\mathbb{R}_{\geqslant 0} \cup \{+\infty\}, \min, +, +, \min, +, \min)$
Minimal cybersecurity skill level	**None** (**0**): no cybersecurity-related skills required **Basic** (**1**): requires basic cybersecurity knowledge and skills **Advanced** (**2**): requires employing advanced cybersecurity-related skills, e.g., executing a MiTM attack on a protocol **Expert** (**3**): requires employing cybersecurity-related skills available to few experts, e.g., return-oriented programming or fault attack on AES **Impossible** ($+\infty$): beyond the known capability of today's human beings	$(\{0, 1, 2, 3, +\infty\}, \min, \max, \max, \min, \max, \min)$
Minimal technical skill level	**None** (**0**): no technical skills required **Basic** (**1**): requires basic technical skills, e.g., finding information online **Advanced** (**2**): requires advanced technical skills, available for graduates of technical vocational schools **Expert** (**3**): requires technical skills available to experienced engineers **Impossible** ($+\infty$): beyond the known capability of today's human beings	$(\{0, 1, 2, 3, +\infty\}, \min, \max, \max, \min, \max, \min)$
Minimal social skill level	**None** (**0**): does not involve social interactions **Basic** (**1**): requires basic social interactions, e.g., obtaining information via a conversation **Advanced** (**2**): requires convincing or tricking someone into doing something they would not do otherwise **Expert** (**3**): requires convincing or tricking someone into doing something punishable by law **Impossible** ($+\infty$): beyond the known capability of today's human beings	$(\{0, 1, 2, 3, +\infty\}, \min, \max, \max, \min, \max, \min)$
Minimal time	**Instantaneous** (**0**): can be performed by the actor in less than a minute, **Quick** (**10**): can be performed by the actor in less than an hour, but not less than a minute **Slow** (**10^2**): can be performed by the actor in less than a week, but not less than an hour **Very slow** (**10^3**): can be performed by the actor in less than six months, but not less than a week **Extremely slow** (**10^4**): can be performed by the actor within a human lifetime, but not less than six months **Impossible** ($+\infty$): not doable within a human lifetime	$(\{0, 10, 10^2, 10^3, 10^4, +\infty\}, \min, \max, \max, \min, \max, \min)$

of the assignment process can be found in [15]. Since for our tree the bound on the size of the set semantics provided by Lemma 5 is small, we compute Pareto optimal values, presented in Table IV, using evaluation on the set semantics.

One can draw several conclusions from Table IV. First, since the optimal values obtained for scenarios S_1 and S_3 are the same, it follows that implementation of only the countermeasure $\mathtt{d}_4$ does not help with countering the Pareto optimal attacks. Second, the values obtained for scenarios S_2 and S_4 imply that implementing both $\mathtt{d}_4$ and $\mathtt{d}_7$ is as effective as implementing $\mathtt{d}_7$ only. Third, even if the defender performs actions $\mathtt{d}_4$, $\mathtt{d}_5$, and $\mathtt{d}_7$ (scenario S_5), most of the Pareto optimal strategies achieving the root goal in scenario S_1 are still available to the attacker. Knowing the values corresponding to strategies that achieve the root goal in particular scenarios, as well as the capabilities of the attacker and constraints on available resources, might help a security expert in making an informed decision on which security measures should be implemented. Finally, Table IV illustrates also the importance of the multi-parameter analysis: in all of the scenarios there is an uncountered Pareto optimal strategy of non-zero cost. This strategy could have been overlooked if the tree was analyzed only with respect to the cost attribute.

Using our Python implementation, the computation of both Pareto optimal values and the corresponding strategies took no more than 0.05 seconds, for each of the five scenarios. Together with the results presented in Section V-B, this suggests that our approach is extremely efficient for analysis of ADTrees corresponding to real-world scenarios.

B. Practical evaluation

To verify that the quantitative analysis of ADTrees using Pareto attribute domains is applicable for trees describing even more complex scenarios than the one from Section V-A, we have tested our implementation on a number of automatically generated trees. Full description of the experimental setup, as well as all the sources necessary to reproduce the results are available at https://github.com/wwidel/pareto-tests. The tests have been performed on a Windows machine running Intel Core i7–5600U CPU at 2.60 GHz dual core with 16 GB of

Table VI: Basic assignment for the actions of the proponent in the case study of power meter tampering.

Basic action of the proponent	**Minimal cost**	**Minimal cybersecurity skill level**	**Minimal technical skill level**	**Minimal social skill level**	**Minimal time**
acquire information from dumpster diving	0	0	0	0	1000
acquire information from public Internet source	0	0	1	0	100
bribe technician to reconfigure the power meter	500	0	0	3	10
bribe technician to reveal power meter credentials	300	0	0	2	10
buy optical probe	71.2	0	1	0	100
coerce technician into reconfiguring the power meter	0	0	0	3	100
coerce technician into revealing power meter credentials	0	0	0	3	10
collect information by exchanging gossips with employees	0	0	0	1	1000
enter power meter credentials	0	0	0	0	0
extract credentials	0	0	1	0	10
find and download software for hacking power meters	0	1	1	0	10
get employed as field technician	0	0	2	1	1000
get employed as intern by the energy provider	0	0	1	1	1000
have physical access to the power meter	0	0	0	0	0
intercept credentials	0	2	1	0	0
locate encrypted credentials in the dump	0	2	2	0	100
make the data dump from hardware component	0	1	3	0	100
make optical probe	14	0	2	0	100
monitor communications between hardware components	0	1	2	0	100
perform brute force attack	0	1	2	0	100
provide power meter credentials	0	0	0	0	0
reconfigure power meter using authorized software/tools	0	0	1	0	10
reconfigure power meter using unauthorized software	0	0	2	0	10
select technician for obtaining power meter credentials	0	0	0	0	100
select technician for reconfiguring power meter	0	0	0	0	100
technician reconfigures power meter using authorized software/tools	0	0	0	0	10
trick technician into revealing power meter credentials	0	0	0	2	100
use optical probe to establish connection to the meter via the optical port	0	0	1	0	10

Table VII: Basic assignment for the actions of the opponent in particular scenarios of the case study of power meter tampering, with $\mathbf{0} = \{(0,0,0,0,0)\}$ and $+\boldsymbol{\infty} = \{(+\infty,+\infty,+\infty,+\infty,+\infty)\}$ (see Remark 1).

Basic action of the opponent	S_1	S_2	S_3	S_4	S_5
$\mathtt{d}_1$ =enforce policy of using strong passwords	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathtt{d}_2$ = enforce policy to minimize Internet disclosure	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathtt{d}_3$ = enforce policy to minimize leakage of physical artefacts	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathtt{d}_4$ = limit the number of possible invalid authentication attempts	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$+\infty$
$\mathtt{d}_5$ = password authentication for establishing connection	$\mathbf{0}$	$\mathbf{0}$	$+\infty$	$+\infty$	$+\infty$
$\mathtt{d}_6$ = require authentication for introducing changes in power consumption configuration	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathtt{d}_7$ = thorough background check before hiring new employees	$\mathbf{0}$	$+\infty$	$\mathbf{0}$	$+\infty$	$+\infty$
$\mathtt{d}_8$ =track popular social engineering attacks and warn personnel	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$

RAM. The main goal of our experiment was to compare how the two methods perform, depending on the characteristics of the analyzed trees. An excerpt from the obtained results is presented in Table VIII.

For a tree T with n nodes and k repeated basic actions of the proponent, two Pareto domains were considered. Each of them is induced by m domains for *minimal cost*, one domain for *minimal difficulty*, and one domain for *minimal time*. Basic assignments β were constructed under the assumption that the opponent performs all of their actions. Values assigned to the basic actions of the proponent were generated randomly. For the computation of Pareto frontiers, the naive method, where each element of a set is compared with the other elements, coordinate by coordinate, was used. We have measured the time of the computation of the Pareto optimal values using the evaluation on the set semantics $\mathrm{Par}_{\mathcal{S}}(T,\beta)$ (which includes the time needed for the construction of the set semantics itself) and using the method of [2] applied to Pareto domains. Each time value presented in Table VIII is an average over twenty measurements.

Table VIII is partitioned into three parts. For the trees from the first part, the performance of the two methods is comparable. For the trees presented in the second part, the computation on the set semantics outperforms the method of [2], while the opposite is true for the third part of the table.

We would like to point out the following observations.

1) The attack trees from the second part of Table VIII have small set semantics, while having a significant number of repeated basic actions.
2) The trees *tree10* and *tree13* have large set semantics, while having a very low number of repeated basic actions.
3) The running times for trees *tree12* and *tree30* differ

significantly, while the two trees have the same number of nodes and repeated basic actions, and small set semantics. However, there are more Pareto optimal values under the basic assignments generated for *tree30*. This illustrates the impact of the actual *values assigned to the basic actions*, which translates into different numbers of Pareto optimal values, on the running time.

VI. Related work

We limit this section to the approaches concerned with multi-parameter quantitative evaluation and ADTrees with repeated labels. The goal is to compare our framework with other existing techniques. For a more exhaustive description of the state of the art on attack tree-based modeling and related tools the reader is referred to [25], [26], and [27].

One way of addressing the problem of multi-parameter quantitative analysis of security using attack tree-based models is to construct an attribute being a combination of several relevant elementary parameters. An example of such an attribute is the *expected outcome* of the attacker, considered by Jürgenson and Willemson in the context of attack trees in [28]. The *expected outcome* represents monetary profit of the attacker, expressed in terms of the *gain* of the attacker in case the attack succeeds, the *costs* of the attack, its *success probability*, as well as the *probability of being caught*, and the related *penalties*. This work uses Boolean functions as the underlying formal model of attack trees. The *expected outcome*'s value is computed for all valuations satisfying the Boolean function representing an attack tree, and the solution with the highest value is retained as the outcome that the attacker can get from performing an attack. Since the logical operators used by Boolean functions are idempotent, the treatment of repeated labels in [28] is similar to the one that we admit in the current work. However, due to the necessity of checking all relevant valuations, the complexity of the solution from [28] is higher than the complexity of our framework.

In [29], Edge et al. discuss how to combine the *probability*, *expected cost*, and *impact* parameters into a metrics called *risk*. The individual parameters are propagated using the standard bottom-up approach, and the risk at each node of an attack tree is then computed according to the formula $(probability/cost) \cdot impact$. A simple analysis of the bottom-up propagation rules for *probability* and *cost* used in [29] implies that they are not suited for trees where a basic action contributes to several attacks.

More recently, several approaches exploiting model checking techniques have been proposed to address the problem of multi-parameter quantitative evaluation on attack tree-based models. The focus of Aslanyan and Nielson in [30] is on attack trees with the *exact cost* and the *probability* parameters. Attack trees are transformed into Markov decision processes with reward structure, and erPCTL[5] queries, such as "what is the *maximum probability* of an attack with the *cost* at most c?" are answered using probabilistic model checking. Compared to our framework, the approach developed by Aslanyan and Nielson deals with two-parameter evaluation (*exact cost* and *probability*) only and similarly to [29], it does not seem to be suited for attack trees containing repeated labels.

In [10], Kumar et al. consider attack trees with basic actions decorated with cost structures modeling *time*, *skills*, *damage*, and *difficulty*. Attack trees are translated into priced timed automata which are then given to the Uppaal Cora model checker where they are queried for quantitative properties of interest expressed with weighted CTL queries. The objective is to provide an effective way of computing the necessary resources (e.g., *time*, *skills*) and the corresponding attack paths leading to the achievement of the root goal. This solution allows the authors to deal with two-parameter optimization using an iterative procedure. The method is suitable for attack trees with repeated basic actions, but does not tackle evaluation on ADTrees nor the probability attribute. In his Ph.D. thesis [31], Kumar automatizes this procedure with the help of ATTop [32], but does not provide time measurements. Interestingly, for the attack tree considered in [32], having 12 nodes, 2 elements in the set semantics, and no repeated basic actions, the authors state that ATTop needed more than 6 seconds for computing an attack of minimal time, i.e., for performing the first step of the iterative method for determining Pareto optimal attacks. In the light of the results presented in Section V, it thus seems that our solution outperforms the method of [10] (on inputs suitable for both methods).

Timed automata-based model checking has also been applied for multi-parameter analysis of ADTrees. In [8] and [9], the authors encode the attacker's and the defender's behavior as a network of timed automata in order to evaluate quantitative queries dependent on *time*, e.g., determining the *probability* of a successful attack or the *expected cost* of succeeding within a given *time*. The evaluation is ensured by the Uppaal model checker. Contrary to the approach presented in [10], the attacker of [8] may try executing their actions several times, with a certain probability. Whereas the Uppaal-based approach of [8] is tailored to specific attributes, namely *cost*, *probability*, and *time*, the solution we propose can be applied to a wide class of attributes whose domains satisfy the assumptions of Theorem 3.

Model checking of ADTrees decorated with the *cost* of attempted execution and the *success probability* is also the focus of Aslanyan et al. in [11]. ADTrees are translated into stochastic two-player games and one- or multi-parameter queries (expressed in probabilistic alternating-time temporal logic with rewards), such as, "is there an attack with the *expected cost* at most c and the *success probability* at least p?" are answered using the PRISM-games tool. In addition, the model checking algorithms implemented in PRISM-games allow to obtain a Pareto curve illustrating the trade-off between the attack *success probability* and its *expected cost* over possible strategies of the attacker. To capture temporal or causal dependencies between the goals of the actors, sequential conjunctive and sequential disjunctive refinements have been added to ADTrees to complement the two standard refinements

[5] erPCTL stands for probabilistic computation tree logic with exact rewards.

Table VIII: Running times of the methods for some trees with n nodes and k repeated basic actions of the proponent.

Parameters							**Time in sec**	
Name of file storing T	n	k	$\lvert\mathcal{S}(T)\rvert$	$\lvert\mathcal{S}(T)\rvert$ bound of Lemma 5	m	Number of Pareto optimal values	$\text{Pareto}_{\mathcal{S}}(T,\beta)$	Method of [2]
tree04	31	7	352	1024	1	3	0.02	0.02
					5	20	0.11	0.05
tree08	37	9	928	4096	1	2	0.07	0.09
					5	30	0.69	0.23
tree12	43	11	2436	16384	1	1	0.27	0.4
					5	72	4.97	3.2
tree20	36	4	832	1024	1	2	0.04	0.01
					5	12	0.07	0.01
tree29	41	10	640	1280	1	2	0.03	0.25
					5	304	13.32	65.05
tree30	43	11	704	1408	1	3	0.03	0.67
					5	184	6.52	67.51
tree31	45	12	768	1536	1	2	0.04	1.12
					5	128	2.92	53.58
tree32	47	13	832	1664	1	4	0.05	3.47
					5	378	27.88	827.92
tree03	31	4	640	1024	1	2	0.04	< 0.01
					5	131	2.77	0.14
tree10	43	2	14336	16384	1	3	9.68	< 0.01
					5	658	2178.31	1.7
tree13	46	0	32768	32768	1	5	81.93	< 0.01
					5	2151	> 3600	5.56
tree24	50	8	9536	16384	1	2	2.9	0.07
					5	15	3.36	0.1

OR and AND. The expressive power of ADTrees from [11] is thus richer than in the case of our work, however, from the perspective of quantitative analysis, our framework is more general, because [11] is limited to the evaluation of two specific attributes, namely *expected cost* and *success probability*.

Amongst all existing solutions for multi-parameter evaluation on ADTrees, the approach introduced by Aslanyan and Nielson in [33] is the closest to our framework. To the best of our knowledge, this is the only work considering Pareto optimization on ADTrees using the bottom-up approach. The authors of [33] propose an ad hoc framework for determining optimal ways of achieving the root goal, with the basic actions being assigned a *probability of successful execution* and a vector of real-valued non-negative *costs*. Their procedures have been developed for trees that do not contain repeated basic actions. The advantages of our framework over the approach of [33] are that, first, it allows for computing strategies that optimize a number of different parameters, and second, that it can be applied to ADTrees with repeated basic actions.

VII. Conclusion

The main objective of the presented work was to develop an efficient method for multi-parameter optimization of security based on ADTrees. The proposed Pareto attribute domains are suitable for this purpose, and can be used with ADTrees containing repeated basic actions. Our construction shows that the multi-parameter evaluation can be addressed with techniques existing for the single-parameter evaluation. Additionally, Theorem 3 constitutes a general algebraic result that might be of independent interest on its own.

We focused on optimization from the point of view of the attacker only. However, the optimization from the point of view of the defender, or both actors at the same time is also worth investigating. As stated in Remark 1, the basic assignments that we consider for the defender are limited to express whether actions are executed or not, without taking their actual values, e.g., *cost*, *probability*, etc. into account. Given the assignment of a number of attributes to the basic actions of the attacker, as well as the *cost* of the basic actions of the defender, which countermeasures should the defender (having a fixed budget) implement to make the attack as "difficult" as possible, in the sense of Pareto optimality? If the actions of the defender are decorated with several attributes, how to determine a Pareto optimal solution to the above problem? How to adapt the efficient methods developed in [2] for ADTrees with repeated actions to probabilistic computations when not only the execution of the countermeasures is considered, but also their success probability? All these problems are worthwhile from the practical perspective and we will explore them in the future.

References

[1] B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer, "Attack–defense trees," *J. Log. Comput.*, vol. 24, no. 1, pp. 55–87, 2014.

[2] B. Kordy and W. Wideł, "On quantitative analysis of attack–defense trees with repeated labels," in *POST*, ser. LNCS, vol. 10804. Springer, 2018, pp. 325–346.

[3] B. Schneier, "Attack trees," *Dr. Dobb's journal*, vol. 24, no. 12, pp. 21–29, 1999.

[4] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *ICISC*, ser. LNCS, vol. 3935. Springer, 2005, pp. 186–198.

[5] National Electric Sector Cybersecurity Organization Resource (NESCOR), "Analysis of selected electric sector high risk failure scenarios, version 2.0," 2015, http://smartgrid.epri.com/doc/NESCOR%20Detailed%20Failure%20Scenarios%20v2.pdf.

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

[7] A. Bossuat and B. Kordy, "Evil twins: Handling repetitions in attack–defense trees — A survival guide," in *GraMSec@CSF*, ser. LNCS, vol. 10744. Springer, 2017, pp. 17–37.

[8] O. Gadyatskaya, R. R. Hansen, K. G. Larsen, A. Legay, M. C. Olesen, and D. B. Poulsen, "Modelling Attack–defense Trees Using Timed Automata," in *FORMATS*, ser. LNCS, vol. 9884. Springer, 2016, pp. 35–50.

[9] R. R. Hansen, P. G. Jensen, K. G. Larsen, A. Legay, and D. B. Poulsen, "Quantitative evaluation of attack defense trees using stochastic timed automata," in *GraMSec@CSF*, ser. LNCS, vol. 10744. Springer, 2017, pp. 75–90.

[10] R. Kumar, E. Ruijters, and M. Stoelinga, "Quantitative Attack Tree Analysis via Priced Timed Automata," in *FORMATS*, ser. LNCS, vol. 9268. Springer, 2015, pp. 156–171.

[11] Z. Aslanyan, F. Nielson, and D. Parker, "Quantitative verification and synthesis of attack-defence scenarios," in *CSF*. IEEE Computer Society, 2016, pp. 105–119.

[12] B. Kordy and W. Wideł, "How well can I secure my system?" in *iFM*, ser. LNCS, vol. 10510. Springer, 2017, pp. 332–347.

[13] B. Kordy, S. Mauw, and P. Schweitzer, "Quantitative Questions on Attack–Defense Trees," in *ICISC*, ser. LNCS, vol. 7839. Springer, 2012, pp. 49–64.

[14] A. Bagnato, B. Kordy, P. H. Meland, and P. Schweitzer, "Attribute Decoration of Attack–Defense Trees," *IJSSE*, vol. 3, no. 2, pp. 1–35, 2012.

[15] B. Fila and W. Wideł, "Attack–defense trees for abusing optical power meters: A case study and the OSEAD tool experience report," 2019, to appear in GraMSec'19.

[16] M. Geilen, T. Basten, B. D. Theelen, and R. Otten, "An Algebra of Pareto Points," *Fundam. Inform.*, vol. 78, no. 1, pp. 35–74, 2007.

[17] A. Buldas, A. Lenin, J. Willemson, and A. Charnamord, "Simple Infeasibility Certificates for Attack Trees," in *IWSEC*, ser. LNCS, vol. 10418. Springer, 2017, pp. 39–55.

[18] W. Wideł. (2018–) `adtrees` Python package. Accessed: 2019-05-06. [Online]. Available: https://github.com/wwidel/adtrees

[19] P. Kelly-Detwiler. (2013) Electricity Theft: A Bigger Issue Than You Think. Accessed: 2019-02-20. [Online]. Available: https://www.forbes.com/sites/peterdetwiler/2013/04/23/electricity-theft-a-bigger-issue-than-you-think/#5475872972ef

[20] B. Krebs. (2012) FBI: Smart Meter Hacks Likely to Spread. Accessed: 2019-02-20. [Online]. Available: https://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread/

[21] N. G. LLC. (2014) World Loses $89.3 Billion to Electricity Theft Annually, $58.7 Billion in Emerging Markets. Accessed: 2019-02-20. [Online]. Available: https://www.prnewswire.com/news-releases/world-loses-893-billion-to-electricity-theft-annually-587-billion-in-emerging-markets-300006515.html

[22] D. C. Weber. (2012) Optiguard: A Smart Meter Assessment Toolkit. Accessed: 2019-02-20. [Online]. Available: https://media.blackhat.com/bh-us-12/Briefings/Weber/BH_US_12_Weber_Eye_of_the_Meter_WP.pdf

[23] M. Carpenter. (2009) Advanced Metering Infrastructure Attack Methodology. Accessed: 2019-02-20. [Online]. Available: http://docshare.tips/ami-attack-methodology_5849023fb6d87fd2bb8b4806.html

[24] J. McCullough. (2010) Deterrent and detection of smart grid meter tampering and theft of electricity, water, or gas. Accessed: 2019-02-20. [Online]. Available: https://www.elstersolutions.com/assets/downloads/WP42-1010A.pdf

[25] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Computer Science Review*, vol. 13-14, pp. 1–38, 2014.

[26] W. Wideł, M. Audinot, B. Fila, and S. Pinchinat, "Beyond 2014: Formal methods for attack tree-based security modeling," 2019, accepted for publication at *ACM Computing Surveys*.

[27] J. B. Hong, D. S. Kim, C. Chung, and D. Huang, "A survey on the usability and practical applications of Graphical Security Models," *Computer Science Review*, vol. 26, pp. 1–16, 2017.

[28] A. Jürgenson and J. Willemson, "Computing exact outcomes of multi-parameter attack trees," in *OTM Conferences (2)*, ser. LNCS, vol. 5332. Springer, 2008, pp. 1036–1051.

[29] K. S. Edge, G. C. Dalton II, R. A. Raines, and R. F. Mills, "Using Attack and Protection Trees to Analyze Threats and Defenses to Homeland Security," in *MILCOM*. IEEE, 2006, pp. 1–7.

[30] Z. Aslanyan and F. Nielson, "Model checking exact cost for attack scenarios," in *POST*, ser. LNCS, vol. 10204. Springer, 2017, pp. 210–231.

[31] R. Kumar, "Truth or dare: Quantitative security risk analysis via attack trees," Ph.D. dissertation, University of Twente, The Netherlands, 2018.

[32] R. Kumar, S. Schivo, E. Ruijters, B. M. Yildiz, D. Huistra, J. Brandt, A. Rensink, and M. Stoelinga, "Effective Analysis of Attack Trees: a Model-Driven Approach," in *FASE*, ser. LNCS, A. Russo and A. Andy Schürr, Eds., vol. 10802. Springer, 2018, pp. 56–73.

[33] Z. Aslanyan and F. Nielson, "Pareto efficient solutions of attack-defence trees," in *POST*, ser. LNCS, vol. 9036. Springer, 2015, pp. 95–114.