On Orderings in Security Models

Paul D. Rowe

The MITRE Corporation, Bedford, MA, USA
prowe@mitre.org

Abstract. Security decisions are often made on the basis of a comparison of two or more alternatives. Is it better go with design A or design B? Which security policy is best for my needs? What combination of defensive mitigations provide the best protection from attack? Implicit in such comparisons are ordering relations $\preceq$ among the alternatives. Such ordering relations crop up in numerous security formalisms. This paper studies preorders that arise in three formalisms for very different domains of security: attack trees, Copland specifications of layered attestations, and cryptographic protocols. While these three areas of study appear to be very different in subject matter and form, we identify a common construction for defining preorders that arise in them. This new perspective unlocks novel connections that should allow insights in one domain to bear fruit in the others as well.

Keywords: Attack Trees · Layered Attestation · Cryptographic Protocols · Security Orderings.

This paper is dedicated to Joshua Guttman in gratitude for what he has taught me. He has helped me to become a better researcher and to search out the essence of an idea. He has also taught me the importance of non-total orderings! This paper is presented in that spirit.

1 Introduction

When applying formal methods to the security of systems, we often want to know if one solution is "better" than another along some dimension of interest. For example, when designing a cryptographic protocol, we may wonder whether design D_1 is better than D_2 in the sense that any security goals achieved by D_2 can also be achieved by D_1 [13]. Similarly, we might want to compare strategies for distributing firewall policies to various network routers and endpoints against their ability to enforce certain prohibitions on patterns of network traffic [1]. In such cases, what we seek is an ordering relation $\preceq$ that captures some aspect of the security characterisitcs of the objects it orders.

It is too much to expect to find a total order. The multidimensional nature of security means that tradeoffs exist between alternatives that generally prevent two arbitrary objects from necessarily being ordered. We thus often content ourselves with preorders, or sometimes partial orders, along various dimensions

of security. Recall that a preorder is a relation $\leqslant$ that is reflexive and transitive, while a partial order is also anti-symmetric ($a \leqslant b$ and $b \leqslant a$ implies $a = b$).

In this work we explore preorders that have been defined for numerous security formalisms and begin to develop a unifying lens through which to view them. This line of investigation began when we identified some surprising parallels between the syntax and semantics of two formalisms. Sequential attack trees [7, 6] allow researchers to formally express ways in which an adversary might attack a system, accounting for disjunction, conjunction, and sequencing of atomic actions. Copland [11] is a specification language for layered attestation defining how to orchestrate integrity measurements of a target system. In developing Copland, we were faced with the following research question: How can we devise an order on Copland expressions that reflects their strength or trustworthiness in the presence of an active attacker? The similarity between attack trees and Copland suggested that we might be able to directly translate prior work on ordering attack trees [6] to the domain of Copland. As we will see below, such a direct translation, while possible, does not produce an order that reflects the trustworthiness of Copland expressions. It does, however, define orders that capture potentially useful performance characteristics of executing Copland expressions.

Since the direct translation does not shed light on trust properties of Copland expressions, we ultimately took a different approach for Copland trust analysis [14]. We eventually understood this approach to defining a preorder to be an instance of a general construction. The original preorder on attack [6] and its direct translation to Copland expressions are also instances of this general construction. In working through the details of these connections, we realized that yet another security-related preorder—one in the domain of cryptographic protocols, and developed by the author with Guttman and Liskov [13]—might also be viewed as an instance of this general construction.

Summary and contributions. The fundamental observation of this paper is that preorders arise naturally out of considering homomorphisms between semantic sets. While this observation is not new in itself, it provides a common vocabulary with which to describe preorders in three domains with drastically different semantics. When the semantics of some formal object (e.g. attack tree, Copland phrase, cryptographic protocol) is given as a set of structures that come equipped with a notion of a homomorphism, we can define preorders on the objects *without reference to the details of the semantics*. That is, we can treat a semantic operator $[\![\cdot]\!]$ as a black box that produces sets of structures. We can then define preorders on objects according to what homomorphisms exist between their semantic sets.

In defining a preorder for sequential attack trees, Horne et al. [6] give a "white-box" treatment of their semantics, and intersperse upward and downward closures under homomorphisms to build preorders. Our first contribution is to reformulate their ideas so we can treat the semantics as a black box that produces sets of graphs. We can then take downward and upward closures of the results without worrying about how the sets of graphs are generated (The-

orems 1 and 2). We then show how to reinterpret the relationships that emerge after taking downward and upward closures in terms of the homomorphisms that exist between the sets produced by the semantics (Theorem 3). This is an alternate way of deriving the attack tree semantics of [6] (Corollary 1) that gives us a general construction for defining preorders from a black-box, base semantics.

Copland phrases bear striking syntactic similarities with attack trees that manifest as structural similarities in their semantics. The general construction suggested by Corollary 1 leads to a direct translation of the preorder on attack trees to a preorder on Copland expressions. The structural similarities in their semantics allows us to translate results from the analysis of attack trees to the analysis of Copland expressions (Theorem 4, Corollary 2) telling us what kind of properties are reflected by the translated preorder. Unfortunately, this translated preorder doesn't capture trust properties of Copland expressions. However, by modifying the Copland base semantics to account for possible adversary actions, our general preorder construction yields an order that *does* capture important aspects of trustworthiness (Theorem 5).

Finally, we demonstrate the generality of our construction of preorders by shifting our focus to the domain of cryptographic protocols. We summarize the strength order of cryptographic protocols that the author defined with Guttman and Liskov [13] and argue that it coincides with our general construction for preorders given the base semantics defined by the protocol analyzer CPSA. Numerous details prevent us from rigorously proving this correspondence, so we record it as Conjecture 1.

The paper is structured as follows. We first present some preliminary definitions and lemmas in Section 2. We then treat attack trees in Section 3, showing how to turn the white-box semantics into a black-box one that allows us to define a general construction of preorders. We introduce Copland in Section 4, and demonstrate the syntactic and semantic similarities with sequential attack trees. In Section 5 we show to leverage that connection to extract useful performance attributes along which to compare Copland phrases. We then alter the Copland base semantics to obtain a trust ordering in Section 6. Finally, in Section 7, we argue that the protocol strength ordering in an instance of our general construction.

2 Preliminaries

The common thread among all the formalisms we consider here is that they pertain to graphs. While some of the structures are graphs with extra information, the core of the structure is still a graph. We therefore focus the types of graphs and homomorphisms between them that will interest us in the current study.

Definition 1 (Graph). *A* directed, labeled, acyclic graph $G = (N, E, \ell)$ *is a triple in which* N *is a finite set of nodes,* $E \subseteq N \times N$ *is a finite set of edges (represented as ordered pairs of nodes from* N*), and* $\ell : N \to L$ *is a labeling function from nodes to some set* L *of labels. Furthermore, the edge relation is*

acyclic. When we use the unqualified term graph*, the qualifiers "directed, labeled, and acyclic" are implied unless otherwise stated.*

Definition 2 (Homomorphism). *A* (graph) homomorphism $\eta : G \to H$ *between graphs* $G = (N_G, E_G, \ell_G)$ *and* $H = (N_H, E_H, \ell_H)$ *is a function* $\eta : N_G \to N_H$ *on the nodes such that for every edge* $(n_1, n_2) \in E_G$*,* $(\eta(n_1), \eta(n_2)) \in E_H$*, and for every node* $n \in N_G$*,* $\ell_G(n) = \ell_H(n)$*.*

A homomorphism is injective *iff the underlying map on nodes is injective. A* smoothing homomorphism *is one which is bijective on nodes.*

Homomorphisms between graphs bestow a preorder on graphs as follows: $G \leqslant H$ if and only if there is some homomorphism $\eta : G \to H$. In fact, any class of structures that admit homomorphisms will bestow a preorder in the natural way. We will rely on this later when we consider graphs with "extra structure". If we only allow injective homomorphisms, then the preorder is actually a partial order (up to isomorphism) because injective homomorphisms in both directions between (finite) graphs G and H imply that G and H are isomorphic.

The homomorphism preorder on graphs admits the standard notions of up-sets (or order filters) and down-sets (or order ideals) [2].

Definition 3 (Up-/down-sets). *Given a preorder* $(\mathcal{P}, \leqslant)$*, a set* $\mathcal{S} \subseteq \mathcal{P}$ *is an* up-set *(or* order filter*) iff for all structures* G *and* H*, whenever* $G \in \mathcal{S}$ *and* $G \leqslant H$*, then* $H \in \mathcal{S}$*.* $\mathcal{S}$ *is a* down-set *(or* order ideal*) iff for all structures* G *and* H*, whenever* $H \in \mathcal{S}$ *and* $G \leqslant H$*, then* $G \in \mathcal{S}$*.*

The upward closure *of a set* $\mathcal{S}$ *is* $\phi(\mathcal{S}) = \{H \in \mathcal{P} \mid \exists G \in \mathcal{S} \wedge G \leqslant H\}$*. Similarly the* downward closure *of a set* $\mathcal{S}$ *is* $\iota(\mathcal{S}) = \{G \in \mathcal{P} \mid \exists H \in \mathcal{S} \wedge G \leqslant H\}$*.*

The symbols ϕ and ι reflect the terminology of order filters and order ideals. Since an important aspect of the present work is to connect with Horne et al.'s work [6], it is important to note that they use an order that is dual to the homomorphism preorder. The result is that their notions of "up" and "down" are reversed from the use in this paper; so their order filters are our order ideals, etc. Readers familiar with [6] will have to swap ϕ and ι when translating between the papers. Of course, the duality principle for ordered sets ensures such translations are possible and meaningful. Despite these challenges of translation, we prefer to work in the homomorphism preorder because homomorphisms are a central, unifying theme across all the formalisms we study here.

We are now ready to define a few operations on graphs that allow us to build new graphs from old ones. They are not new and can already be found in [6].

Definition 4 ($\uplus, *$). *Let* $G = (N_G, E_G, \ell_G)$ *and* $H = (N_H, E_H, \ell_H)$ *be graphs. Then we can define*

- $N = N_G \times \{0\} \cup N_H \times \{1\}$
- $E = \{((x, 0), (y, 0)) \mid (x, y) \in E_G\} \cup \{((x, 1), (y, 1)) \mid (x, y) \in E_H\}$
- $\ell(n, 0) = \ell_G(n)$ *and* $\ell(n, 1) = \ell_H(n)$*.*

N is the disjoint union of the nodes of G and H, E is the disjoint union of the edges of G and H, and ℓ is the natural labeling of the nodes in N inherited from ℓ_G and ℓ_H.

We call the graph (N, E, ℓ) the disjoint union *of G and H, denoted $G \uplus H$.*

If we additionally define $E' = E \cup ((N_G \times \{0\}) \times (N_H \times \{1\}))$, then we call the graph (N, E', ℓ) the sequential composition *of G and H, denoted $G * H$.*

We can easily lift these two operations on graphs into two corresponding operations on *sets* of graphs in the following way.

Definition 5 ($\bowtie, \rightsquigarrow$). *If $\mathcal{S}_1$ and $\mathcal{S}_2$ are sets of graphs, then the* distributive product *$\mathcal{S}_1 \bowtie \mathcal{S}_2$ is defined by $\{G_1 \uplus G_2 \mid G_1 \in \mathcal{S}_1 \wedge G_2 \in \mathcal{S}_2\}$. The* pointwise sequential composition *of two sets of graphs $\mathcal{S}_1 \rightsquigarrow \mathcal{S}_2$ is defined by $\{G_1 * G_2 \mid G_1 \in \mathcal{S}_1 \wedge G_2 \in \mathcal{S}_2\}$.*

We now prove a few properties about how upward and downward closures distribute over the above graph operations.

Lemma 1. *For any sets of labeled digraphs $\mathcal{S}$ and $\mathcal{T}$, the following equalities hold.*

$$\begin{aligned}\iota(\mathcal{S} \cup \mathcal{T}) &= \iota(\mathcal{S}) \cup \iota(\mathcal{T}) \\ \iota(\mathcal{S} \bowtie \mathcal{T}) &= \iota(\mathcal{S}) \bowtie \iota(\mathcal{T}) \\ \iota(\mathcal{S} \rightsquigarrow \mathcal{T}) &= \iota(\iota(\mathcal{S}) \rightsquigarrow \iota(\mathcal{T}))\end{aligned}$$

Proof. We only prove here the most interesting of the equations. The other two proofs are quite similar.

$$\begin{aligned}\iota(\mathcal{S} \rightsquigarrow \mathcal{T}) &= \{G \mid \exists S \in \mathcal{S}, T \in \mathcal{T}, G \leqslant S * T\} \\ &= \{G \mid \exists G_1, G_2, G_1 \leqslant S, G_2 \leqslant T, G \leqslant G_1 * G_2\} \\ &= \{G \mid \exists G_1 \in \iota(\mathcal{S}), \exists G_2 \in \iota(\mathcal{T}), G \leqslant G_1 * G_2\} \\ &= \iota(\iota(\mathcal{S}) \rightsquigarrow \iota(\mathcal{T}))\end{aligned}$$

□

Lemma 2. *For any sets of graphs $\mathcal{S}$ and $\mathcal{T}$, the following equalities hold.*

$$\begin{aligned}\phi(\mathcal{S} \cup \mathcal{T}) &= \phi(\mathcal{S}) \cup \phi(\mathcal{T}) \\ \phi(\mathcal{S} \bowtie \mathcal{T}) &= \phi(\phi(\mathcal{S}) \bowtie \phi(\mathcal{T})) \\ \phi(\mathcal{S} \rightsquigarrow \mathcal{T}) &= \phi(\phi(\mathcal{S}) \rightsquigarrow \phi(\mathcal{T}))\end{aligned}$$

Proof. The proof is similar to the proof of Lemma 1 and so is omitted. □

3 Attack Trees

Attack trees [15] are a popular way for security experts to formalize their thought process about how an adversary might attack a system. They allow an analyst to express combinations of activities an adversary may or must perform in order to successfully attack a system. In their original formulation, the leaves of attack trees were labeled with adversary activities, and the internal nodes were labeled with attacker sub-goals. Two types of branching were defined: disjunctive branching in which satisfying any of the child nodes is sufficient to satisfy the parent, and conjunctive nodes in which all children must be satisfied in order for the parent node to be satisfied. More recently, Jhawar et al. [7] have introduced a sequence node to attack trees in which all the children must be satisfied *in the given order* for the parent node to be satisfied. This allows attack trees to capture causal or dependency relationships among the actions. Such *sequential attack trees* are the object of study in this section.

A full treatment of attack trees in general, and sequential attack trees in particular, is out of scope for this work. For a more comprehensive introduction to all types of attack trees, we direct the reader to a useful survey by Wideł et al. [17].

A key observation is that the structure of attack trees allows them to be expressed as terms in a grammar in which internal nodes of the tree are represented by an operator corresponding to the intended semantics of satisfaction as described above. That is, we can build up attack trees out of internal nodes labeled by one of the following three operators: $\triangledown, \triangle, \triangleright$ representing disjunction, conjunction, and sequence, respectively. They satisfy the following grammar:

$$T :: \; A \;\mid\; T \triangledown T \;\mid\; T \triangle T \;\mid\; T \triangleright T \tag{1}$$

where A represents a set of atomic actions. In practice we can allow the operators to have arity greater than 2, as is done in [7], however it is more convenient for our purposes (and without loss of generality) to use this more restricted syntax.

Numerous semantic interpretations have been given to this syntax, but we focus on the series-parallel graph semantics in which the meaning of an attack tree is given as a set of series-parallel graphs. The original semantics for sequential attack trees given in [7] uses graphs with labeled edges instead of labeled nodes. We follow the presentation in [6] and consider a dual notion of series-parallel. As discussed in [16], series parallel graphs as defined below are precisely the line graphs of so-called two-terminal series-parallel graphs as used in [7].

Definition 6 (Series-parallel). *A* series-parallel graph *over a set of possible nodes N is defined inductively as follows.*

- *A single labeled node is a series-parallel graph.*
- *If G and H are series-parallel graphs then $G \uplus H$ is a series-parallel graph.*
- *If G and H are series-parallel graphs then $G * H$ are both series parallel graphs*

The original semantics of [7] associates to any sequential attack tree a *set* of series-parallel graphs. The transitive closure of a series-parallel graph defines a partially ordered set. The idea of the semantics is that node represent atomic events which are ordered in the induced partially ordered set if one event depends on the results of another. Disjunction in attack trees results in several possibilities requiring a set of graphs. Thus the union in the following definition is a union of sets of graphs (not the disjoint union $\uplus$ of graphs).

Definition 7 (Base semantics). *The* base semantics *for attack trees is defined inductively as follows, where N_a denotes the graph with a single node whose label is a.*

$$[\![a]\!]_{\mathcal{B}} = \{N_a\} \qquad [\![t_1 \triangledown t_2]\!]_{\mathcal{B}} = [\![t_1]\!]_{\mathcal{B}} \cup [\![t_2]\!]_{\mathcal{B}}$$
$$[\![t_1 \vartriangle t_2]\!]_{\mathcal{B}} = [\![t_1]\!]_{\mathcal{B}} \bowtie [\![t_2]\!]_{\mathcal{B}} \qquad [\![t_1 \triangleright t_2]\!]_{\mathcal{B}} = [\![t_1]\!]_{\mathcal{B}} \rightsquigarrow [\![t_2]\!]_{\mathcal{B}}$$

The base semantics in Def. 7 was designed to determine *equivalence* of attack trees. That is, two trees are equivalent precisely when they have the same semantics. But when this semantics was introduced in [7], no attention was paid to distinguishing the strength of attack trees.

To address this questions of relative strength, Horne et al. [6] introduced two additional semantics for sequential attack trees that create a "specialization" preorder on them. These preorders correspond closely to two variations on the base semantics, one involving *down-sets* and the other involving *up-sets* of graphs.[1]

Definition 8 (Down-set semantics). *The* down-set semantics *for attack trees is given by the following.*

$$[\![a]\!]_{\mathcal{I}} = \{N_a\} \qquad [\![t_1 \triangledown t_2]\!]_{\mathcal{I}} = [\![t_1]\!]_{\mathcal{I}} \cup [\![t_2]\!]_{\mathcal{I}}$$
$$[\![t_1 \vartriangle t_2]\!]_{\mathcal{I}} = [\![t_1]\!]_{\mathcal{I}} \bowtie [\![t_2]\!]_{\mathcal{I}} \qquad [\![t_1 \triangleright t_2]\!]_{\mathcal{I}} = \iota([\![t_1]\!]_{\mathcal{I}} \rightsquigarrow [\![t_2]\!]_{\mathcal{I}})$$

Definition 9 (Up-set semantics). *The* up-set semantics *is given by the following:*

$$[\![a]\!]_{\mathcal{F}} = \phi(\{N_a\}) \qquad [\![t_1 \triangledown t_2]\!]_{\mathcal{F}} = [\![t_1]\!]_{\mathcal{F}} \cup [\![t_2]\!]_{\mathcal{F}}$$
$$[\![t_1 \vartriangle t_2]\!]_{\mathcal{F}} = \phi([\![t_1]\!]_{\mathcal{F}} \bowtie [\![t_2]\!]_{\mathcal{F}}) \qquad [\![t_1 \triangleright t_2]\!]_{\mathcal{F}} = \phi([\![t_1]\!]_{\mathcal{F}} \rightsquigarrow [\![t_2]\!]_{\mathcal{F}})$$

In these definitions, we only apply the downward (respectively upward) closures whenever the combining operator does not produce an down-set (respectively up-set). Applying them in the other cases would be redundant. Since we do not restrict ourselves to smoothing homomorphisms, we must close under ϕ in more cases than is needed for the corresponding semantics in [6]. It is interesting that this difference resulted in no change for our down-set semantics.

[1] Recall that we are working in an order that is dual to the one used in [6]. In comparing with that work, the reader must substitute ι with ϕ (and vice versa) and similarly for $\mathcal{I}$ and $\mathcal{F}$.

These two semantics generate two natural preorders on attack trees:

$$\begin{aligned} t_1 \preceq_{\mathcal{I}} t_2 \text{ iff } [\![t_1]\!]_{\mathcal{I}} \subseteq [\![t_2]\!]_{\mathcal{I}} \\ t_1 \preceq_{\mathcal{F}} t_2 \text{ iff } [\![t_1]\!]_{\mathcal{F}} \subseteq [\![t_2]\!]_{\mathcal{F}} \end{aligned} \tag{2}$$

The purpose of these preorders is to enable quantitative comparisons among attack trees. It is possible to make assertions about quantitative comparisons using only the two preorders used above, provided the quantitative measures are sound with respect to the preorders. The details of such comparisons (including a definition of soundness) are given in Section 5. In the meantime, we proceed with an alternative derivation of the preorders in Eqn. 2.

Specialization using $[\![\cdot]\!]_{\mathcal{B}}$. Definitions 8 and 9 interleave the graph operations with the downward and upward closure operations. Our first novel insight regarding these two semantics is that they are equivalent to first applying the base semantics of Def. 7, then applying either the downward or the upward closure.

Theorem 1. *For any attack tree t, $[\![t]\!]_{\mathcal{I}} = \iota([\![t]\!]_{\mathcal{B}})$.*

Proof. We proceed by induction on the structure of t. We start with the case in which the tree is an atom a. $[\![a]\!]_{\mathcal{I}} = \{N_a\}$. And $[\![a]\!]_{\mathcal{B}} = \{N_a\} = \iota(\{N_a\})$. When $t = t_1 \triangleright t_2$ we have

$$\begin{aligned} [\![t_1 \triangleright t_2]\!]_{\mathcal{I}} &= \iota([\![t_1]\!]_{\mathcal{I}} \rightsquigarrow [\![t_2]\!]_{\mathcal{I}}) \\ &= \iota(\iota([\![t_1]\!]_{\mathcal{B}}) \rightsquigarrow \iota([\![t_2]\!]_{\mathcal{B}})) \\ &= \iota([\![t_1]\!]_{\mathcal{B}} \rightsquigarrow [\![t_2]\!]_{\mathcal{B}}) \\ &= \iota([\![t_1 \triangleright t_2]\!]_{\mathcal{B}}) \end{aligned}$$

where the second equality is the inductive hypothesis and the third equality is by Lemma 1.The other inductive cases follow analogously from Lemma 1. □

The analogous result holds for the up-set semantics.

Theorem 2. *For any attack tree t, $[\![t]\!]_{\mathcal{F}} = \phi([\![t]\!]_{\mathcal{B}})$.*

Proof. The proof uses the same ideas as that of Theorem 1 and so is omitted. □

In a sense, Theorems 1 and 2 show that the downward and upward closures in Defs. 8 and 9 are needlessly entangled with aspects of the syntax of attack trees. The base semantics of Def. 7 provides a natural and clear interpretation for attack trees. Instead of messing with the internal structure of that semantics to extract information about specialization, we can first compute the base semantics $[\![t]\!]_{\mathcal{B}}$ and then compute either the downward or upward closure.

While the semantics of [6] generate finite sets because they restrict themselves to smoothing homomorphisms that do not add any new nodes to graphs, we have chosen to consider arbitrary homomorphisms which means that the upward closure is an infinite set. This introduces a new challenge not faced

in [6]. Namely, to determine if $t_1 \preceq_{\mathcal{F}} t_2$ we must devise a procedure for deciding whether $\phi([\![t_1]\!]_{\mathcal{B}}) \subseteq \phi([\![t_2]\!]_{\mathcal{B}})$ that does not require us to compute either set. It turns out we can easily do this by considering the homomorphisms that exist between $[\![t_1]\!]_{\mathcal{B}}$ and $[\![t_2]\!]_{\mathcal{B}}$. This allows us to reduce an infinite question of set membership to a finite questions about what homomorphisms exists among two finite sets of graphs.

Definition 10 (Supports, Covers). *Given two sets of graphs $\mathcal{S}$ and $\mathcal{T}$, we say that $\mathcal{S}$* supports *$\mathcal{T}$ iff for every $H \in \mathcal{T}$, there is some $G \in \mathcal{S}$, such that $G \leqslant H$. We say that $\mathcal{T}$* covers *$\mathcal{S}$ iff for every $G \in \mathcal{S}$ there is some $H \in \mathcal{T}$ such that $G \leqslant H$.*

Intuitively, $\mathcal{S}$ supports $\mathcal{T}$ if $\mathcal{S}$ is big enough to contain sources of homomorphisms to everything in $\mathcal{T}$. Similarly, $\mathcal{T}$ covers $\mathcal{S}$ if $\mathcal{T}$ is big enough to contain targets of homomorphism from everything in $\mathcal{S}$.

Theorem 3. *For any two sets of graphs $\mathcal{S}$ and $\mathcal{T}$, $\iota(\mathcal{S}) \subseteq \iota(\mathcal{T})$ if and only if $\mathcal{T}$ covers $\mathcal{S}$. Similarly, $\phi(\mathcal{S}) \subseteq \phi(\mathcal{T})$ if and only if $\mathcal{T}$ supports $\mathcal{S}$.*

Proof. Suppose $\iota(\mathcal{S}) \subseteq \iota(\mathcal{T})$. We have $\mathcal{S} \subseteq \iota(\mathcal{S}) \subseteq \iota(\mathcal{T}) = \{G \mid \exists H \in \mathcal{T}, G \leqslant H\}$. So, for every $G \in \mathcal{S}$, there is some $H \in \mathcal{T}$ such that $G \leqslant H$. But that's precisely the definition of $\mathcal{T}$ covers $\mathcal{S}$.

Now suppose that $\mathcal{T}$ covers $\mathcal{S}$. Then, for every $G \in \mathcal{S}$, there is some $H \in \mathcal{T}$ such that $G \leqslant H$. Now let $K \in \iota(\mathcal{S})$, so that there is some $G \in \mathcal{S}$ such that $K \leqslant G$. But from above, there is some $H \in \mathcal{T}$ such that $G \leqslant H$. Transitivity of $\leqslant$ gives us $K \leqslant H$, and hence $K \in \iota(\mathcal{T})$. Since K was chosen abritrarily from $\iota(\mathcal{S})$ we conclude that $\iota(\mathcal{S}) \subseteq \iota(\mathcal{T})$ as required.

Now suppose that $\phi(\mathcal{S}) \subseteq \phi(\mathcal{T})$. We have $\mathcal{S} \subseteq \phi(\mathcal{S}) \subseteq \phi(\mathcal{T}) = \{H \mid \exists G \in \mathcal{T}, G \leqslant H\}$. So for all $H \in \mathcal{S}$ there is some $G \in \mathcal{T}$ such that $G \leqslant H$. But this is the definition of $\mathcal{T}$ supports $\mathcal{S}$ as required.

Finally, suppose that $\mathcal{T}$ supports $\mathcal{S}$. So, for every $H \in \mathcal{S}$, there is some $G \in \mathcal{T}$ such that $G \leqslant H$. Now consider $K \in \phi(\mathcal{S})$. By definition, there is some $H \in \mathcal{S}$ such that $H \leqslant K$. But from above, there is some $G \in \mathcal{T}$ such that $G \leqslant H$. Using the transitivity of $\leqslant$ we find $G \leqslant K$, showing that $K \in \phi(\mathcal{T})$. Since K was chosen arbitrarily from $\phi(\mathcal{S})$, we conclude $\phi(\mathcal{S}) \subseteq \phi(\mathcal{T})$ as required. □

Theorem 3 gives us an effective procedure for resolving any question of the form $t_1 \preceq_{\mathcal{F}} t_2$ or $t_1 \preceq_{\mathcal{I}} t_2$. We simply compute $[\![t_1]\!]_{\mathcal{B}}$ and $[\![t_2]\!]_{\mathcal{B}}$ and enumerate the homomorphisms that exists between elements of those finite sets to determine if one of them covers or supports the other.

Interestingly, while Horne et al. were not faced with this challenge, they nevertheless devised a procedure for resolving such questions that does not amount to a direct check of set inclusion between finite sets. Instead, they develop two additional semantics into an extension of a fragment of linear logic (called MAV [5]) and proving that two trees can be ordered precisely when the linear logic interpretation of one implies the interpretation of the other. Since MAV is decidable, they can extract a decision procedure. This logical encoding is reminiscent of prior

work by the author with Guttman and Liskov [13]. In that work, we developed a method for comparing the strength of cryptographic protocols by assigning them formulas in first order logic (expressing the security goals they satisfy) and ordering them according to implication. We will say more about this connection in Section 7.

An immediate corollary of Theorem 3 is the following theorem that says we can recover the intended preorders on attack trees without explicit reference to the downward and upward closures.

Corollary 1. *For any two attack trees t_1 and t_2 we have the following.*

$$t_1 \preceq_{\mathcal{I}} t_2 \text{ iff } [\![t_2]\!]_{\mathcal{B}} \text{ covers } [\![t_1]\!]_{\mathcal{B}}$$
$$t_1 \preceq_{\mathcal{F}} t_2 \text{ iff } [\![t_2]\!]_{\mathcal{B}} \text{ supports } [\![t_1]\!]_{\mathcal{B}}$$

This corollary gives us a reusable recipe for generating preorders. If we are given a class of objects, and some semantic operator $[\![\cdot]\!]$ on those objects yielding sets of graphs, we can *define* two preorders $\preceq_{\mathcal{I}}$ and $\preceq_{\mathcal{F}}$ in terms of which semantic sets cover or support which others. In fact, since the notions of covering and supporting are well defined for any structures that admit homomorphisms, this construction is quite general.

Throughout the rest of the paper, we repeatedly take inspiration from Corollary 1 to define new orders. For structures other than attack trees, we identify a "base" semantics playing the same role as Def. 7, and then define preorders according to which sets in those semantics cover or support which others. When the base semantics has a structural connection with $[\![\cdot]\!]_{\mathcal{B}}$ (such as Copland, presented in the next section) we will see that we can transport some results from attack trees to the new setting. However, it is important to note that this construction works even when the base semantics bears no resemblance to $[\![\cdot]\!]_{\mathcal{B}}$, and we will explore two such instances in Sections 6 and 7.

4 Copland

In this section we turn our attention to Copland, a domain-specific language for specifying layered attestations [11]. On the surface, Copland has little to do with attack trees. However, we will describe a surprisingly deep connection between the two formalisms that allows some results about the preorders on attack trees to be applied directly to Copland specifications. We also believe research into attack trees can benefit from insights established about Copland.

Remote attestation is a technique for establishing trust in the integrity of a remote system. This is done by having agents local to the target system measure various aspects of the target. This typically involves hashing portions of a component's memory with predictable values that are likely to be changed as a result of an attack to the component. The measurement evidence gathered from various subcomponents is then bundled together both to reflect the way in which it was collected (who measured what, and in what order), and to provide integrity protection to the evidence itself so it cannot be tampered with in

transit. Layered attestations leverage hierarchical dependencies built into many modern systems to ensure trust in the measurement apparatus can be established before relying on it to establish trust in the target. Copland was designed to support flexible specifications of layered attestations, and connect to a trust analysis framework [12] (about which more will be said below).

What follows is a very brief overview of the syntax and semantics of Copland. The reader should consult [11, 4] for more in-depth descriptions and motivations. An expression in Copland is called a *phrase*. The syntax of Copland phrases is given by the following grammar:

$$
\begin{array}{lll}
C :: A(V) & & \text{Atomic action with arguments} \\
\quad \mid C \ \rightarrow \ C & & \text{Linear sequence} \\
\quad \mid C \ \overset{\pi}{\prec} \ C & & \text{Sequential branching} \\
\quad \mid C \ \overset{\pi}{\sim} \ C & & \text{Parallel branching} \\
\quad \mid @_P \ [\ C \ ] & & \text{At place} \\
\quad \mid (\ C \ ) & & \text{Grouping}
\end{array}
$$

Copland is parameterized by the set of atomic actions available to use. The syntax is designed to specify both the control flow of actions as well as the data flow of evidence among them. The control flow operators are similar to the operators used in attack trees. Copland contains two sequential operators ($\rightarrow, \prec$), and one conjunction operator ($\sim$). The purpose of having two distinct sequential operators is to define distinct data flow patterns for the sequential control flow. This will manifest in the semantics given below. Copland does not contain any disjunction operators. There is no fundamental barrier to including disjunction; it simply was not immediately relevant for the intended use of Copland phrases. Copland also contains a new type of operator $@_P$. It indicates the transfer of data and control from one entity to another. The decorators π above $\overset{\pi}{\prec}$ and $\overset{\pi}{\sim}$ specify fine grained aspects of data flow that do not affect the results of this paper, so we will say no more about them.

The semantics of Copland is reminiscent of the base semantics of Def.7 for sequential attack trees from [7], but it is significantly complicated by the need to carefully track data flow. As with attack trees, Copland semantics is also given in terms of series-parallel graphs, but it relies on an auxiliary *evidence-type semantics* that defines how the type of evidence is transformed throughout the execution of a phrase. In addition to a Copland phrase c, this evidence-type semantics, denoted $\mathcal{E}(c, p, e)$, is sensitive to the place p currently in control of the execution and to the evidence type e built up so far. The details of this semantics is not relevant to our current study, so we treat it as a black box that returns a given type of evidence. The semantics associated with some of the operators includes some "extra" events (req, rpy, split, and join) that serve to coordinate the evidence-type semantics with the data flow. In contrast with Def. 7, the Copland semantics results in a single graph, not a set of graphs, so it uses the graph constructors from Def. 4, and not Def. 5.

Definition 11 (Copland semantics). *The* Copland semantics *for a Copland phrase is a graph defined by the following.*

$$
\begin{aligned}
[\![a(\bar{v})]\!]_p^e &= N_a(\bar{v}, p, e) \\
[\![@_q\, c]\!]_p^e &= \mathsf{req}(p,q) * [\![c]\!]_q^e * \mathsf{rpy}(p,q) \\
[\![c_1 \to c_2]\!]_p^e &= [\![c_1]\!]_p^e * [\![c_2]\!]_p^{\mathcal{E}(c_1,p,e)} \\
[\![c_1 \overset{\pi}{\prec} c_2]\!]_p^e &= \mathsf{split}(p,\pi) * [\![c_1]\!]_p^{\pi_1(e)} * [\![c_2]\!]_p^{\pi_2(e)} * \mathsf{joins}(p) \\
[\![c_1 \overset{\pi}{\sim} c_2]\!]_p^e &= \mathsf{split}(p,\pi) * ([\![c_1]\!]_p^{\pi_1(e)} \uplus [\![c_2]\!]_p^{\pi_2(e)}) * \mathsf{joinp}(p)
\end{aligned}
$$

There is enough detail in Def. 11 to warrant a more detailed comparison with attack tree base semantics from Def. 7. Notice first that, since the event semantics relies on the evidence-type semantics it is also parameterized by the current entity in control p and the input evidence type e denoted by sub- and superscripts on the semantics operator. The semantics carefully transforms these values in recursively evaluating the semantics of sub-phrases. Nothing of this sort exists in the attack tree semantics because data flow is not accounted for. As mentioned above, the data flow is the key differentiator between Copland's two sequential operators. With $c_1 \to c_2$, c_2 is evaluated with the evidence produced by c_1. By contrast, in $c_1 \overset{\pi}{\prec} c_2$, c_2 is evaluated with $\pi_2(e)$ which is derived from the evidence type built up before c_1 and c_2 are sequenced.

The "extra" events, such as $\mathsf{req}(p,q)$, $\mathsf{split}(p,\pi)$ etc., are essential for keeping the series-parallel graph semantics in sync with the evidence-type semantics. However, these events to not alter the fundamental way in which the semantics of the sub-phrases are connected. Namely, sequential operators use the sequential composition of graphs, and the conjunction operator uses the disjoint union of graphs. The primary difference in these connections is that Copland does not use the corresponding $\bowtie$ and $\rightsquigarrow$ operators which work on sets of graphs. This is entirely due to the absence of a disjunctive operator in Copland. The result is that the semantics of a given phrase is a single graph instead of a set of graphs. In fact, we could easily re-write the Copland semantics to work on sets of graphs using $\bowtie$ and $\rightsquigarrow$, but as the resulting sets would be singletons, there is no advantage to doing so beyond clarifying the connection to attack tree semantics.

Based on these observations, the following table depicts a rough correspondence between Copland operators and attack tree operators. As each side has features not captured by the other, it is not a simple bijection. Furthermore, details regarding data flow mean there is not an exact equivalence in the semantics of corresponding operators. Nevertheless, this table represents a surprisingly deep connection between the two formalisms, especially considering they were developed independently.

The correspondence is strong enough to suggest leveraging the results from Section 3 to obtain two preorders on Copland phrases. After all, the Copland semantics was not designed with strength comparison in mind, just as was the case with the original semantics for sequential attack trees. A naive approach would be to attempt to replicate the semantics from Def. 8 and 9. But it is not

Table 1. Correspondence between Copland and Attack Tree operators.

Copland	Attack Trees
$a(\bar{v})$	a
$@_q\, c$	
$c_1 \rightarrow c_2$	$t_1 \triangleright t_2$

Copland	Attack Trees
$c_1 \overset{\pi}{\prec} c_2$	$t_1 \triangleright t_2$
$c_1 \overset{\pi}{\sim} c_2$	$t_1 \triangle t_2$
	$t_1 \triangledown t_2$

immediately obvious how to interleave the upward and downward closures with the series-parallel graph operations. Taking inspiration from Corollary 1, we can avoid taking upward and downward closures altogether and define two preorders on Copland phrases as follows:

Definition 12 (Copland preorders). *The two preorders $\preceq^C_{\mathcal{I}}$ and $\preceq^C_{\mathcal{F}}$ on Copland phrases are defined as follows.*

$$c_1 \preceq^C_{\mathcal{I}} c_2 \text{ iff } \{[\![c_2]\!]^e_p\} \text{ covers } \{[\![c_1]\!]^e_p\}$$
$$c_1 \preceq^C_{\mathcal{F}} c_2 \text{ iff } \{[\![c_2]\!]^e_p\} \text{ supports } \{[\![c_1]\!]^e_p\}$$

Since the Copland semantics produces a single series-parallel graph, this is equivalent to:

$$\begin{aligned} c_1 \preceq^C_{\mathcal{I}} c_2 \text{ iff } [\![c_1]\!]^e_p \leqslant [\![c_2]\!]^e_p \\ c_1 \preceq^C_{\mathcal{F}} c_2 \text{ iff } [\![c_2]\!]^e_p \leqslant [\![c_1]\!]^e_p \end{aligned} \tag{3}$$

Notice that, since the Copland semantics produces a single series-parallel graph, $c_1 \preceq^C_{\mathcal{I}} c_2$ iff $c_2 \preceq^C_{\mathcal{F}} c_1$. This is not true in general for semantics that result in sets of graphs.

5 Attribute Domains

In this section we demonstrate that the connection between attack trees and Copland is not a superficial similarity. The syntactic correspondence identified in the previous section allows us to transport results about attack trees to results about Copland phrases. In particular, we focus on how the preorders of Corollary 1 and Def. 12 relate to quantitative comparisons using attribute domains.

Definition 13 (Attribute domain). *An* attribute domain *is a tuple $D = (V, f_1, \ldots, f_n)$ where V is a set of values ordered by $\leqslant$, and $f_1, \ldots, f_n$ are functions associated with a set of operators $o_1, \ldots, o_n$. An* attribute *is a pair (D, ν) where D is an attribute domain and $\nu : A \rightarrow V$ is a function from the set of basic actions to the set of values.*

When applied to attack trees or Copland phrases, attribute domains provide a way of giving them quantitative values, assuming a base function $\nu : A \rightarrow V$ is given. The value $\mathcal{V}$ for an attack tree or a Copland phrase is defined inductively as follows:

$$\mathcal{V}_\nu(a) = \nu(a) \qquad \mathcal{V}_\nu(t_1\ o_i\ t_2) = f_i(\mathcal{V}_\nu(t_1), \mathcal{V}_\nu(t_2))$$

where o_i is an operator, and f_i is its associated function.

Since the order of functions matters in the definition of an attribute domain, we fix an order for the operators of attack trees and Copland phrases respectively. For attack tree attribute domains, the list of functions (f_1, f_2, f_3) will correspond to the list $(\triangledown, \triangle, \triangleright)$, in that order. For Copland attribute domains, the list of functions (f_1, f_2, f_3, f_4) will correspond to the list $(\overset{\pi}{\sim}, \rightarrow, \overset{\pi}{\prec}, @_q)$, in that order.

Definition 14 (Soundness). *An attribute domain D is* sound *with respect to a given preorder $\preceq$ if and only if either*

- *for all t_1, t_2, ν, $t_1 \preceq t_2$ implies $\mathcal{V}_\nu(t_1) \leqslant \mathcal{V}_\nu(t_2)$, or*
- *for all t_1, t_2, ν, $t_1 \preceq t_2$ implies $\mathcal{V}_\nu(t_2) \leqslant \mathcal{V}_\nu(t_1)$.*

In the former case we call it co-variantly sound*, in the latter case it is* contra-variantly *sound.*

Notice that soundness is not a bi-conditional. Completeness would involve the reverse implication. But since many examples of interest involve using sets of values V that are totally ordered, and since the preorders on attack trees and Copland phrases are only preorders, we should not expect completeness in most cases.

Horne et al. [6] identify four attribute domains that are sound with respect to $\preceq_{\mathcal{I}}$ and $\preceq_{\mathcal{F}}$. These are presented in Table 2.

Table 2. Some sound attribute domains for attack trees.

Attribute domain	Preorder	Soundness Direction	Interpretation
(ℕ,min,+,max)	$\preceq_{\mathcal{I}}$	Contra-variant	Minimum experts required
(ℝ,min,max,+)	$\preceq_{\mathcal{F}}$	Contra-variant	Minimum attack time
(ℕ,max,+,max)	$\preceq_{\mathcal{F}}$	Co-variant	Guards needed to counter attack
(ℝ,max,max,+)	$\preceq_{\mathcal{I}}$	Co-variant	Time required for all attacks

The first attribute domain can represent the minimum number of experts required to attack the system. This is like a measure of parallelism. If two actions can be done in parallel, then two distinct experts will be required to take advantage of this parallelism. So this is a measure of the minimal parallelism allowed by any attack. The second row can represent the minimum time required to perform an attack. The third row is sort of dual to the first row in that it essentially measures the maximal amount of parallelism of any attack. This could correspond to the number of guards required to be on duty to thwart an attack. Finally, the last row can represent the time required to make all attacks possible.

The correspondence from Table 1 suggests corresponding attribute domains for Copland. By assigning the same functions to corresponding operators, and by interpreting $@_q$ in such a way that it contributes nothing to the attribute, we immediately get a few attribute domains that are sound for the Copland semantics.

Table 3. Sound attribute domains for Copland phrases.

Attribute domain	Preorder	Soundness Direction
($\mathbb{N}$,+,max,max,0)	$\preceq^C_{\mathcal{I}}$	Contra-variant
($\mathbb{R}$,max,+,+,0)	$\preceq^C_{\mathcal{F}}$	Contra-variant
($\mathbb{N}$,+,max,max,0)	$\preceq^C_{\mathcal{F}}$	Co-variant
($\mathbb{R}$,max,+,+,0)	$\preceq^C_{\mathcal{I}}$	Co-variant

Theorem 4. *The attribute domains in each row of Table 3 are each sound with respect to the corresponding preorder in the indicated direction.*

Theorem 4 can be proved directly, but it is also a consequence of the soundness results shown in Table 2 and the structural semantic connection between attack trees and Copland phrases.

Notice that the four attribute domains for attack trees are collapsed down to two attribute domains for Copland. This is because the attack tree attribute domains differ in pairs only in how disjunction is interpreted. As Copland has no disjunction, the correspondence collapses each pair. In the context of layered attestation, rows 1 and 3 can be interpreted as identifying the number of CPU cores required to take advantage of parallelism. As there is only one graph, the maximum is the same as the minimum, so these collapse to the same attribute domain. Rows 2 and 4 correspond to the time it takes to execute a Copland phrase. This could be interpreted as either the minimum time or the maximum time, depending on the interpretation of the function ν used.

These attribute domains are slightly contrived in the context of Copland because they essentially ignore the extra events that get added in the Copland semantics. We can easily account for these by incorporating values for these extra events into the functions corresponding to the operators that add them. For example, if the events req, rpy, split, and join took at least q, p, s, and j time units to complete, then we would want to consider the attribute domain specified as $(\mathbb{R}, \max_{s+j}, +, +_{s+j}, q+p)$ where $a +_{s+j} b$ is defined to be $s + a + b + j$ and $\max_{s+j}(a, b)$ is defined to be $\max(s + a + j, s + b + j)$. This attribute domain builds in the time for the added events in the natural way. We then easily get two more soundness results.

Corollary 2. *The attribute domain* $(\mathbb{R}, max_{s+j}, +, +_{s+j}, q+p)$ *is co-variantly sound with respect to* $\preceq^C_{\mathcal{I}}$ *and contra-variantly sound with respect to* $\preceq^C_{\mathcal{F}}$.

Proof. It is a simple exercise to verify that the consistent addition of s, j, q, and p to the values does not affect the relative order of the resulting functions. Soundness thus follows immediately from Thm. 4. □

The connection between attack trees and Copland thus provides a way for us to preorder phrases according to certain performance aspects. This is potentially very useful in designing selection policies for layered attestations. There are potentially numerous reasons to prefer one phrase over another, many of which concern the performance profiles. Some of these performance profiles are well

captured by attribute domains sound with respect to $\preceq_{\mathcal{I}}^{C}$ and $\preceq_{\mathcal{F}}^{C}$. However, the direct translation of Table 2 to attribute domains for Copland only yields two attribute domains from the original four. This suggests an opportunity to research other attribute domains that might be relevant to the performance profile in executing a Copland phrase. Are there other dimensions along which we would like to compare Copland phrases that are not captured by attribute domains sound with respect to either ordering?

6 Copland Trust Ordering

Our primary interest in Copland phrases is not their performance aspects such as how quickly they can be executed, i.e., those attributes that correspond to the preorders defined in Section 4. We are more interested in ordering phrases based on how well they convey system trust in the presence of an active adversary.

In this section we apply the recipe for defining preorders suggested by Corollary 1 to a base semantics that incorporates adversary events into the graphs of actions. Since Copland phrases have no syntactic elements corresponding to adversarial actions, this semantics has a much looser connection to the syntactic structure of a phrase. As a consequence, we must contend with the fact that there is no straightforward way to leverage attribute domains as in the previous section.

The base adversarial semantics for Copland phrases derives from our prior work on layered attestations [12] in which we established a suitable adversary model. Concretely, we assume that adversaries can corrupt and repair components at will. Corrupted measurers will fail to detect any corruption in their targets. If the target of a measurement is corrupt before it is measured, then to avoid detection the adversary must either repair the target or corrupt the measurer (or some component the measurer depends on to correctly measure). In this model it is always possible for the adversary to undetectably corrupt a given component. But we can ask, "assuming that some given target was corrupt at the time it was measured, and that the attestation detects no corruptions, what else must the adversary have done to avoid detection?"

We recently developed a tool chain to answer such questions [14]. This tool chain computes all minimal, adversarial executions consistent with the traditional Copland semantics of Def. 11, together with some initial assumptions or hypotheses H. These include assumptions of the form that some set of components are corrupt at the time they are measured, and that all corruptions go undetected.

Let us denote this minimal set computed by the tool chain as $\mathcal{A}_H(t)$, where $\mathcal{A}$ indicates it is an adversary-enriched semantics, and H denotes the particular hypotheses assumed. This is a set of graphs with extra structure to encode assumptions about which components are corrupt at which events. Taking this as our new "base" semantics, we again define new two preorders on Copland phrases. This time they are parameterized by the hypotheses H used in the computation of the semantics.

Definition 15 (Copland trust ordering).

$$c_1 \preceq^H_{\mathcal{I}} c_2 \textit{ iff } \mathcal{A}_H(c_2) \textit{ covers } \mathcal{A}_H(c_1)$$
$$c_1 \preceq^H_{\mathcal{F}} c_2 \textit{ iff } \mathcal{A}_H(c_2) \textit{ supports } \mathcal{A}_H(c_1)$$

Def. 15 is a mechanical application of the recipe suggested by Corollary 1. A key question is whether these preorders correspond to the strength of a Copland phrase, i.e., its ability to accurately convey trust information in the presence of an active adversary. Do either of the preorders in Def. 15 capture a useful notion of trustworthiness? If so, which one?

To better understand the situation, consider the notion of trustworthiness developed in [12]. As mentioned above, the underlying adversary model always admits ways for the adversary to corrupt components without being detected. It can simply corrupt components between the time they are measured and the time they take a measurement. Alternatively, it can corrupt components deep enough in the system to undermine measurements at higher layers. We refer to these two strategies as *recent* or *deep* corruptions, respectively. Thus, recent or deep strategies allow an adversary to go undetected by an attestation. The primary question becomes whether or not the adversary has any other strategies that might be easier to perform. A rough measure of the strength of a Copland phrase is to say that it is strong (or strong enough) if the recent or deep corruption strategies are the only ones that will succeed.

Since the new base semantics $\mathcal{A}_H(\cdot)$ is not inductively defined according to the syntactic structure of a Copland term, we cannot meaningly define attribute domains in the same way as Section 5. However, we can still define natural maps into other ordered sets that clearly correspond to the notion of trust described above. In particular, we can define a mapping RD (for recent or deep) of Copland phrases into the 2-point lattice $\{\bot, \top\}$. $RD(c) = \top$ if the only way for an adversary to avoid detection is by employing recent or deep strategies. $RD(c) = \bot$ if there is some strategy that is neither recent nor deep that still allows the adversary to avoid detection. In fact, this mapping will depend on the set of hypotheses H as described above. Thus, we really have a family of maps $RD_H : \text{Copland} \to \{\bot, \top\}$ and a corresponding family of induced orders $\preceq^H_{RD}$. At a coarse level, then, we consider a Copland phrase c to be sufficiently trustworthy relative to hypotheses H if $RD_H(c) = \top$, and untrustworthy otherwise.

We can now ask whether either of the preorders of Def. 15 capture the notion of trust described above. That is, we can ask if either of $c_1 \preceq^H_{\mathcal{I}} c_2$ or $c_1 \preceq^H_{\mathcal{F}} c_2$ implies the same (or possibly opposite) order on $RD_H(c_1)$ and $RD_H(c_2)$. This would be a type of soundness of RD_H with respect to the preorders.

To investigate this question, consider a simple attestation scenario involving two measurements. Atomic Copland phrase $\mathsf{m}_1(x, y)$ represents the measurement of some component y by a well-protected component x. Atomic phrase $\mathsf{m}_2(y, z)$ represents the measurement of component z by component y. Thus, x represents a "deep" component of the system. There are (at least) three natural ways to

order these measurements.

$$c_1 = \mathsf{m}_1(x, y) \overset{\pi}{\sim} \mathsf{m}_2(y, z) \tag{4}$$

$$c_2 = \mathsf{m}_2(y, z) \overset{\pi}{\prec} \mathsf{m}_1(x, y) \tag{5}$$

$$c_3 = \mathsf{m}_1(x, y) \overset{\pi}{\prec} \mathsf{m}_2(y, z) \tag{6}$$

Using the tool chain we developed in [14] we can compute $\mathcal{A}_H(c_i)$ for $1 \leqslant i \leqslant 3$, where H is the hypothesis that z is corrupt when it is measured. The details of how they are computed are well beyond the scope of this work, but the results are shown in Figures 1–3. The figures show the transitive reduction of the edge relations which are all transitive. This semantics also "forgets" non-measurement events. The events labeled $\mathsf{c}(\cdot)$ (respectively $\mathsf{r}(\cdot)$) represent an event in which the adversary corrupts (respectively repairs) the given component.

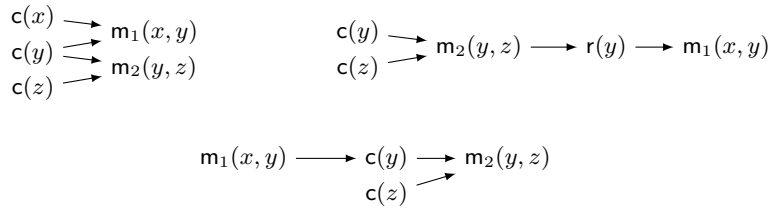

Fig. 1. The three graphs in $\mathcal{A}_H(c_1)$.

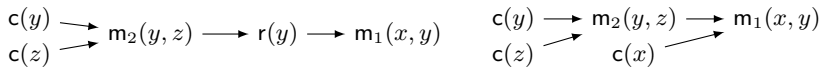

Fig. 2. The two graphs in $\mathcal{A}_H(c_2)$.

Fig. 3. The two graphs in $\mathcal{A}_H(c_3)$.

Phrases c_1 and c_2 each admit executions in which the deep component x is not corrupted, and the corruptions of y and z need not occur recently (e.g.

after the start of the attestation). Thus $RD_H(c_1) = RD_H(c_2) = \bot$. In the executions admitted by c_3, there is either a deep corruption of x, or there is a recent corruption of y. Thus $RD_H(c_3) = \top$.

We can similarly determine which of the sets $\mathcal{A}_H(c_i)$ cover or support which others. It is a simple exercise to check that $\mathcal{A}_H(c_1)$ supports both $\mathcal{A}_H(c_2)$ and $\mathcal{A}_H(c_3)$, and that neither of the latter two support each other. Also, none of the sets covers any of the others. Thus, the only orders involving $\preceq^H_{\mathcal{I}}$ and $\preceq^H_{\mathcal{F}}$ that hold are $c_2 \preceq^H_{\mathcal{F}} c_1$ and $c_3 \preceq^H_{\mathcal{F}} c_1$.

This small investigation suggests that $\preceq^H_{RD}$ is not related to $\preceq^H_{\mathcal{I}}$, but that it might be related (contravariantly) to $\preceq^H_{\mathcal{F}}$. Indeed, we can prove that $\preceq^H_{RD}$ is contravariantly sound with respect to $\preceq^H_{\mathcal{F}}$.

Theorem 5. *If $c_1 \preceq^H_{\mathcal{F}} c_2$ then $c_2 \preceq^H_{RD} c_1$.*

Proof. Since $c_2 \preceq^H_{RD} c_1$ holds for all values of $RD_H(c_1)$ and $RD_H(c_2)$ *except* $RD_H(c_2) = \bot$ and $RD_H(c_1) = \top$, it suffices to show that whenever $c_1 \preceq^H_{\mathcal{F}} c_2$ and $RD_H(c_1) = \top$ then $RD_H(c_2) = \top$ as well.

First note that $RD_H(c_1) = \top$ means that for every $G \in \mathcal{A}_H(c_1)$, G contains a recent corruption or a deep corruption. Also, recent and deep corruptions are both preserved under homomorphisms. Since $c_1 \preceq^H_{\mathcal{F}} c_2$, we know that for every $G_2 \in \mathcal{A}_H(c_2)$, there is some $G_1 \in \mathcal{A}_H(c_1)$ such that $G_1 \preccurlyeq G_2$. But since G_1 has a recent or deep corruption, this is preserved by the homomorphism to G_2. Thus every element of $\mathcal{A}_H(c_2)$ has a recent or deep corruption. So $RD_H(c_2) = \top$. □

Theorem 5 is encouraging. It shows that the generalized up-set (adversary-enriched) semantics for Copland captures an intuitive, and independently defined notion of trust. This works out despite the fact that we are in a setting where the "base" semantics is given as an arbitrary set of structures not explicitly tied to the syntax. In fact, it is encouraging enough to suggest that research in attack trees might benefit from applying such a generalization. For example, attack-defense trees have been proposed as a richer formalism to discuss offensive and defensive strategies for system security. Could there be a semantics in the spirit of the adversary-enriched semantics for Copland that could be leveraged in this way to order attack(-defense) trees? Copland's tracking of dataflow could also be replicated to enrich the semantics of attack trees along that dimension.

Nevertheless, the soundness of Theorem 5 is a little unsatisfying. For one thing, the same soundness does not hold for the strict preorders. This is evident from the fact that $c_2 \prec^H_{\mathcal{F}} c_1$ but $c_1 \not\preceq^H_{RD} c_2$. It is, in some sense, too sensitive to differences in Copland phrases. Just because two phrases are strictly ordered, we cannot conclude that one must force the adversary into recent or deep corruptions. Thus, we can only leverage $c_1 \preceq^H_{\mathcal{F}} c_2$ for our purposes if we know $RD_H(c_2) = \top$ or $RD_H(c_1) = \bot$. The fact that c_2 and c_3 are $\preceq^H_{\mathcal{F}}$-incomparable is also worrisome. Knowing that one phrase forces the adversary into recent or deep corruptions and the other doesn't is not enough to guarantee they will be ordered by $\preceq^H_{\mathcal{F}}$. This indicates that $\preceq^H_{\mathcal{F}}$ is, in some sense, not sensitive enough.

Theorem 5 encourages us to push forward with new ideas for ordering Copland phrases (or other formalisms), but it raises at least as many questions as

it answers. What security aspects is $\preceq_{\mathcal{F}}^{H}$ really capturing beyond the notion of recent or deep adversary strategies? Is soundness enough to view it as a generalization of the 2-point lattice ordering, or do the issues in the previous paragraph undermine that viewpoint? Is there a logical characterization of the content of models found in $\mathcal{A}_H(c)$ that enables soundness with respect to logical implication? We hope the investigation of this paper will spur research along these lines. The generality of the approach enables progress to be made by those working in diverse subfields of formal methods for security.

7 Cryptographic Protocols

While we were spurred to investigate the connection between attack trees and Copland due to the similarities in their underlying syntax and semantics, the results of Section 6 show that the general construction can yield interesting results even in cases with semantics that are utterly unrelated to that of attack trees. Therefore, before concluding, we make a brief detour into the world of cryptographic protocols to demonstrate the generality of the recipe for constructing preorders suggested by Corollary 1.

In 2016, Guttman, together with the author and our colleague Moses Liskov established a methodology for determining a strength order on cryptographic protocols [13]. As we outline below, the preorder generated in this way seems to correspond to the preorder that would arise from the construction we have used repeatedly throughout this paper. Due to space limitations, we can only give a very high-level overview, and the correspondence, while highly suggestive, is still technically conjectural.

The general idea from [13] is to derive a logical formula

$$\mathcal{L}(\Phi, P) = \forall X.(\Phi \Longrightarrow \bigvee_{1 \leqslant i \leqslant n} \exists Y.\Psi_i)$$

that expresses the strongest conclusion achievable by protocol P from hypothesis Φ. For those familiar with CPSA, Φ represents the input to a search, and each Ψ_i represents one of the shapes of protocol P, while X and Y range over events and the variables used in messages. In general, protocols need their own set of predicates to describe their possible executions. That is, a predicate saying that some role of protocol P_1 has executed some number of steps with a given set of parameters will not, in general, have an interpretation in the executions of protocol P_2. However, when the protocols are sufficiently similar, the same logical language can easily apply to both protocols. This allows us to create a preorder on protocols parameterized by the hypothesis Φ: $P_1 \preceq_\Phi P_2$ iff $\mathcal{L}(\Phi, P_2) \Longrightarrow \mathcal{L}(\Phi, P_1)$. This says that P_2 is stronger than P_1 (with respect to Φ) if any goal achieved by P_1 is also achieved by P_2.

It has been shown in [9] that $\mathcal{L}(\Phi, P)$ corresponds to a run of the protocol analyzer CPSA [10] when provided an input $\mathbb{A}$ that corresponds to Φ. CPSA works in the strand spaces model of cryptographic protocols (pioneered by Guttman),

and produces the minimal, essentially different executions of a protocol consistent with some initial assumptions. Concretely, given a skeleton (i.e. partial execution) $\mathbb{A}$ of protocol P, it produces a finite set $\mathcal{S}_{\mathbb{A}}(P)$ of realized skeletons (i.e. full executions) $\mathbb{B}$ for which $\mathbb{A} \preccurlyeq \mathbb{B}$.[2] Furthermore, $\mathcal{S}_{\mathbb{A}}(P)$ supports the set of all realized skeletons $\mathbb{C}$ satisfying $\mathbb{A} \preccurlyeq \mathbb{C}$. That is, for all realized skeletons $\mathbb{C}$ such that $\mathbb{A} \preccurlyeq \mathbb{C}$, there is an element $\mathbb{B} \in \mathcal{S}_{\mathbb{A}}(P)$ such that $\mathbb{B} \preccurlyeq \mathbb{C}$.

The correspondence between $\mathcal{L}(\Phi, P)$ and CPSA's search arises from the ability to associate to every skeleton $\mathbb{A}$ a characteristic formula $\chi(\mathbb{A})$. For certain syntactic classes of formulas Φ we can revert the process to get a characteristic skeleton $\sigma_P(\Phi)$. (The inverse σ_P depends on the protocol because different protocols admit different structures.) For our purposes we may assume these two processes are inverses. Thus, in the previous paragraph, we choose $\mathbb{A}$ to be $\sigma_P(\Phi)$. The correspondence is completed by the fact that $\Psi_i = \chi(\mathbb{B}_i)$ for $\mathbb{B}_i \in \mathcal{S}_P(\mathbb{A})$ [9].

When comparing the strength of two protocols, we start with a common hypothesis Φ. We then translate that hypothesis into (possibly distinct) skeletons $\mathbb{A}_1 = \sigma_{P_1}(\Phi)$ and $\mathbb{A}_2 = \sigma_{P_2}(\Phi)$ of P_1 and P_2 respectively. Applying CPSA, we obtain the two sets of shapes $\mathcal{S}_{\mathbb{A}_1}(P_1)$ and $\mathcal{S}_{\mathbb{A}_2}(P_2)$. We then recover $\mathcal{L}(\Phi, P_i)$ by applying χ to the sets of shapes. This now gives us access to the preorder $\preceq_\Phi$.

One key advantage of converting CPSA's analysis back into logical form is that it allows direct comparison between the results. Due to the detailed message structure that is purposefully not represented in the logical formulas, we typically can't talk about homomorphisms between skeletons of two different protocols. This prevents us from directly applying our recipe for defining preorders that requires us to determine whether $\mathcal{S}_{\mathbb{A}_1}(P_1)$ covers or supports $\mathcal{S}_{\mathbb{A}_2}(P_2)$, or vice versa. The translation into logic acts as a substitute in much the same way that Horne et al. [6] define a translation into linear logic to help them compute the comparisons. However, Guttman has developed a way to convert skeletons of P_1 into skeletons of P_2, provided there is a well-defined protocol transformation $\mathcal{T} : P_1 \to P_2$ [3]. So if $\mathbb{A}_1$ is a skeleton of P_1, then $\mathcal{T}(\mathbb{A}_1)$ is a skeleton of P_2. Furthermore, for sufficiently close protocols, $\chi(\mathbb{A}_1) = \chi(\mathcal{T}(\mathbb{A}_1))$. (For more distantly related protocols, the equality must be downgraded to an equivalence.) Using this theory of protocol transformation we conjecture that the $\preceq_\Phi$ preorder corresponds to one of the preorders generated using Theorem 3.

Conjecture 1. *Suppose that* $\sigma_{P_1}(\Phi) = \mathbb{A}_1$ *and* $\sigma_{P_2}(\Phi) = \mathbb{A}_2$. *Let* $\mathcal{T} : P_1 \to P_2$ *be a well-defined protocol transformation. Then*

$$P_1 \preceq_\Phi P_2 \text{ iff } \mathcal{S}_{\mathbb{A}_2}(P_2) \text{ covers } \mathcal{T}(\mathcal{S}_{\mathbb{A}_1}(P_1)), \text{ and}$$
$$P_2 \preceq_\Phi P_1 \text{ iff } \mathcal{T}(\mathcal{S}_{\mathbb{A}_1}(P_1)) \text{ covers } \mathcal{S}_{\mathbb{A}_2}(P_2).$$

We leave it as a conjecture for now because a treatment that attends to all the details about protocol transformations and conversions to and from logical formulas would require considerable care and is beyond the aims of this paper. Indeed, it is not entirely clear that the bi-implication is correct. Perhaps it only follows that if the semantic sets are in the right covering relationship, then the

[2] For technical reasons, we must restrict ourselves to injective homomorphisms only.

corresponding order holds. Our main purpose is to highlight similarities and differences with the preorders from earlier sections to gain insights into how we might fruitfully generalize the approach to generating preorders.

We conclude with a few remarks about this conjecture that speak to the generality of our construction. Firstly, although, skeletons of a protocol are graph-like, they actually contain more information than is contained in the structures for attack trees or Copland phrases. This demonstrates that the approach is not tied to semantics that use sets of graphs, but can apply to other structures that admit a homomorphism ordering. Additionally, the conjecture would not hold if we restricted attention to smoothing homomorphisms only as is done in [6]. $P_1 \preceq_\Phi P_2$ will hold when the shapes of P_2 can infer the existence of more activity (more nodes of the graph), not just more orderings among events. This was one of the principal drivers for our choice not to restrict ourselves to smoothing homomorphisms in Section 3 which resulted in an alteration to the up-set semantics compared to it counterpart in [6].

Just as in Section 6, the base CPSA semantics is not tightly tied to the syntax of protocols. So, although we cannot easily define attribute domains for protocols, the translation into logic can be viewed as serving a similar purpose. Indeed, because the logical content captures all the needed details, the corresponding order is not only sound, but the conjecture is that it is also complete with respect to the order defined through our construction. If true, this would mean that the preorder constructed according to the methods of this paper precisely capture the intended content of security goals. The connection between $\preceq^H_{RD}$ and $\preceq^H_{\mathcal{F}}$ in Section 6 was much weaker. Perhaps we could identify a sound and complete logical characterization of $\preceq^H_{\mathcal{F}}$ that comes with a clear interpretation in terms of trust.

Finally, notice that the conjecture only uses the notion of "covers" and not the notion of "supports". Thus, $\preceq_\Phi$ is a kind of down-set semantics. That is, it shares the same form as the $\preceq_{\mathcal{I}}$ order on attack trees. If we write it as $\preceq^\Phi_{\mathcal{I}}$, this suggests the natural alternative $\preceq^\Phi_{\mathcal{F}}$ defined according to an up-set semantics. That is, which $P_1 \preceq^\Phi_{\mathcal{F}} P_2$ when $\mathcal{T}(\mathcal{S}_{\mathbb{A}_1}(P_2))$ supports $\mathcal{S}_{\mathbb{A}_2}(P_1)$. It is not immediately clear what this preorder captures. We consider it an open problem to provide a preorder with a natural interpretation that corresponds to (or at least is sound with respect to) $\preceq^\Phi_{\mathcal{F}}$.

8 Conclusion

In this paper we explored numerous security-related preorders from the literature. We developed a way to generalize specialization preorders of attack trees [6] to essentially any formalism which has a set-based denotational semantics for which there exists a notion of homomorphism on elements of the sets. In particular, we defined the two notions of *covers* and *supports*, and showed how these generate a preorder on the semantic sets that corresponds to the up- and down-set semantics of attack trees respectively. We applied this general construction to Copland phrases for layered attestation in two settings. The first is an adversary-

free setting in which the preorders correspond to certain performance aspects of the intended executions. The second is an adversary-enriched setting in which the semantics no longer closely reflects algebraic properties of the syntax. Along the way we identified a similarity to preorders defined on cryptographic protocols. While the details are beyond this paper, we conjectured that the protocol preorders can be viewed as an instance of the general construction used here.

While our focus has been on three formalisms, the results obtained are suggestive that the construction may have a much greater reach. But the current study also raises some questions. The protocol preorder is constructed using the *covers* relation, while the corresponding construction for Copland adversarial semantics requires the *supports* relation. It is not clear when to expect the use of one versus the other. In fact, the *covers* and *supports* notions have been previously identified as providing a basis for constructing powerdomains for programs with non-deterministic execution [8, 18]. A more thorough investigation into the relation of the current study with that past work may shed light on our questions and suggest a more abstract standpoint from which to view security orderings.

Acknowledgments

I would like to thank Ian Kretz and John Ramsdell for our continued collaboration on the topic of layered attestation. This paper arose out of our earlier shared attempt to leverage attack trees to help order Copland phrases.

References

1. Adão, P., Focardi, R., Guttman, J.D., Luccio, F.L.: Localizing firewall security policies. In: 2016 IEEE 29th Computer Security Foundations Symposium (CSF). pp. 194–209 (2016). https://doi.org/10.1109/CSF.2016.21
2. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, 2 edn. (2002). https://doi.org/10.1017/CBO9780511809088
3. Guttman, J.D.: Establishing and preserving protocol security goals. Journal of Computer Security **22**(2), 203–267 (2014). https://doi.org/10.3233/JCS-140499
4. Helble, S.C., Kretz, I.D., Loscocco, P.A., Ramsdell, J.D., Rowe, P.D., Alexander, P.: Flexible mechanisms for remote attestation. ACM Trans. Priv. Secur. **24**(4) (Sep 2021). https://doi.org/10.1145/3470535
5. Horne, R.: The consistency and complexity of multiplicative additive system virtual. Scientific Annals of Computer Science **25**(2), 245–316 (2015). https://doi.org/10.7561/SACS.2015.2.245
6. Horne, R., Mauw, S., Tiu, A.: Semantics for specialising attack trees based on linear logic. Fundam. Informaticae **153**(1-2), 57–86 (2017). https://doi.org/10.3233/FI-2017-1531
7. Jhawar, R., Kordy, B., Mauw, S., Radomirović, S., Trujillo-Rasua, R.: Attack trees with sequential conjunction. In: Federrath, H., Gollmann, D. (eds.) ICT Systems Security and Privacy Protection. pp. 339–353. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-18467-8_23
8. Plotkin, G.D.: A powerdomain construction. SIAM J. Comput. **5**(3), 452–487 (1976). https://doi.org/10.1137/0205035

9. Ramsdell, J.D.: Deducing security goals from shape analysis sentences. CoRR **abs/1204.0480** (2012)
10. Ramsdell, J.D., Guttman, J.D., Liskov, M.D., Rowe, P.D.: The CPSA specification: A reduction system for searching for shapes in cryptographic protocols (2012)
11. Ramsdell, J.D., Rowe, P.D., Alexander, P., Helble, S.C., Loscocco, P., Pendergrass, J.A., Petz, A.: Orchestrating layered attestations. In: Nielson, F., Sands, D. (eds.) Principles of Security and Trust. pp. 197–221. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-17138-4_9
12. Rowe, P.D.: Confining adversary actions via measurement. In: Kordy, B., Ekstedt, M., Kim, D.S. (eds.) Graphical Models for Security. pp. 150–166. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-46263-9_10
13. Rowe, P.D., Guttman, J.D., Liskov, M.D.: Measuring protocol strength with security goals. Int. J. Inf. Sec. **15**(6), 575–596 (2016). https://doi.org/10.1007/s10207-016-0319-z
14. Rowe, P.D., Ramsdell, J.D., Kretz, I.D.: Automated trust analysis of Copland specifications for layered attestation. In: Proceedings of the 23rd International Symposium on Principles and Practice of Declarative Programming. PPDP '21, Association for Computing Machinery, New York, NY, USA (2021)
15. Schneier, B.: Attack trees. Dr. Dobb's Journal **24**(12), 21–29 (1999)
16. Valdes, J., Tarjan, R.E., Lawler, E.L.: The recognition of series parallel digraphs. In: Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing. p. 1–12. STOC '79, Association for Computing Machinery, New York, NY, USA (1979). https://doi.org/10.1145/800135.804393
17. Wideł, W., Audinot, M., Fila, B., Pinchinat, S.: Beyond 2014: Formal methods for attack tree–based security modeling. ACM Comput. Surv. **52**(4) (Aug 2019). https://doi.org/10.1145/3331524
18. Winskel, G.: On powerdomains and modality. Theoretical Computer Science **36**, 127–137 (1985). https://doi.org/https://doi.org/10.1016/0304-3975(85)90037-4