REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
07/21/2021	Book Chapter	

4. TITLE AND SUBTITLE
On Orderings in Security Models

5a. CONTRACT NUMBER
W56KGU-18-D-0004

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Paul D. Rowe

5d. PROJECT NUMBER
0721N6BZMA

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730

8. PERFORMING ORGANIZATION REPORT NUMBER
PRS-21-1785

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
United States Army (ACC-APG)

10. SPONSOR/MONITOR'S ACRONYM(S)
ACC-APG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
preprint to Lecture Notes in Computer Science / Protocols, Strands, and Logic; DOI 10.1007/978-3-030-91631-2_21

14. 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 • 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 defi

15. SUBJECT TERMS
Computer Security; Cryptography; Attack tree; Cryptographic protocols; Different domains; Layered attestation; Ordering relations; Preorders; Security attacks; Security modeling; Security ordering; Security policy; Forestry

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE		22	Susan Carpenito
					19b. TELEPHONE NUMBER *(Include area code)* 781-271-7646

Approved for Public Release; Distribution Unlimited.
Public Release Case Number 21-1628

On Orderings in Security Models

Paul D. Rowe

The MITRE Corporation

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 framework for characterizing and 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.

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

The view, opinions, and/or findings contained in this report are those of The MITRE Corporation and should not be construed as an official Government position, policy, or decision, unless designated by other documentation. This technical data deliverable was developed using contract funds under Basic Contract No. W56KGU-18-D-0004.

them. This line of investigation began when we identified some surprising parallels between the syntax and semantics of sequential attack trees [7,6] and that of Copland phrases for layered attestation [11]. We tried (and failed) to leverage techniques for ordering attack trees [6] to represent the security aspects of layered attestations we wanted to capture. However, those techniques *are* relevant for ordering Copland phrases along non-security-related dimensions (c.f. Section 5). Eventually we took a different approach to the analysis of Copland phrases for layered attestation which is ongoing [14]. We report here the surprising parallels we discovered between these two disparate formalisms, and provide some insights into why they failed.

In the course of working through the detailed correspondence between attack trees and Copland, we hit upon another striking similarity in a family of preorders we previously developed for cryptographic protocols in collaboration with Guttman and Liskov [13]. This observation has caused us to realize we may be able to salvage the connection we first discovered between attack trees and Copland, and incorporate similar techniques into the approach we take in [14].

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 upward and downward closures of the results without worrying about how the sets of graphs are generated (Theorems 1 and 2). We then show how to reinterpret the relationships that emerge after taking upward and downward closures in terms of the homomorphisms that exist between the sets produced by the semantics (Theorem 3).

We then turn to Copland phrases to specify layered attestations. We highlight some syntactic and semantic similarities Copland shares with sequential attack trees. While this observation initially led us to believe we might be able to leverage the white-box approach of [6], we ultimately leverage the black-box approach to define preorders on Copland phrases. We do, however, rely on the syntactic and semantic similarity to convert soundness results from attack trees to Copland. This provides insights into what kinds of attributes are reflected by the newly defined Copland preorders.

Unfortunately, these attributes do not relate to the trustworthiness of a Copland phrase, which relies on an richer semantics that accounts for the ways an adversary might interfere with the execution of a layered attestation. We have developed a way of computing this adversary-enriched semantics by progressively building up larger structures from simpler ones [14]. The process is guided by domain-specific reasoning about an adversary's capabilities. The process is somewhat reminiscent of how our protocol analyzer CPSA [10] analyzes protocols. Observing this similarity offers an opportunity, because we have a way of preordering protocols based on their analysis in CPSA [13]. However, our previous attempts to define a "trust" ordering of Copland phrases based on this observation failed. The preorder on protocols depends on a translation of CPSA results into first-order logic in which the implication preorder nicely captures protocol strength. The analogous translation from our new analysis techniques of Copland turns out not to reflect our intended trust ordering.

Despite these obstacles, we show how one might reinterpret the protocol preordering as an instance of the general constructions of Theorem 3. We express this reinterpretation as a formal conjecture of the equivalence of the protocol preorder and a preorder constructed on the basis of Theorem 3.

Armed with new hope for the generality of the preorder constructions, we return to Copland and investigate how to capture our intuitions about a trust ordering as a preorder constructed with Theorem 3. We show in Theorem 4 that our translation of Copland trust into a simple 2-point lattice is sound with respect to one of the preorders constructed by the methods of this paper.

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. 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 digress to a discussion of the preorder on cryptographic protocols in Section 6, returning to an investigation of the Copland trust ordering in Section 7.

2 Preliminaries

The common thread among all the formalisms we consider here is that they pertain to graphs. While some of the structures (like skeletons) 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. All homomorphisms in this paper are assumed to be injective unless otherwise stated. We write* $G \cdot\!\!\to H$ *to indicate that there exists an injective homomorphism* $\eta : G \to H$.

Homomorphisms (even non-injective ones) between graphs bestow a preorder on graphs as follows: $G \leqslant H$ if and only if there is some homomorphism $\eta : H \to G$. (The reason for "reversing" the direction of $\leqslant$ from the direction of the homomorphism will be made clear below.) In fact, any class of structures that admit homomorphisms will bestow a preorder in the same way. We will rely on this later when we consider graphs with "extra structure".

The homomorphism preorder on graphs allows us to talk about up-sets and down-sets.

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

The symbols ι and ϕ reflect the terminology of order ideals and order filters. We hasten to note that, in order theory, an ideal (resp. filter) is not the same as an order ideal (resp. order filter). Ideals and filters satisfy additional conditions about the existence of meets and joins (see, e.g. [2]). We stick with the more standard terminology of up-sets and down-sets but retain the suggestive notation as it is used in [6].

This connection to order filters and order ideals explains our choice of "reversing" the direction of $\leqslant$ relative to homomorphisms. In order to use the same notation as [6] and respect the typical directions of order filters and order ideals, we are forced to have $G \leqslant H$ correspond to a homomorphism $\eta : H \to G$. This is rather confusing if you are used to considering the "information preorder" on structures in which homomorphisms—which add information—move upwards in the preorder. Indeed, in the remainder of the paper there will be several concepts that cause the reader to "flip" which direction is up due to contravariant operations. Such dizzying gymnastics are liable to cause vertigo! Unfortunately it cannot be avoided. The reader is warned. For ease of comprehension we will generally prefer to write $G \cdot\!\!\to H$ instead of $H \leqslant G$.

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 ($\uplus, *$). *If* $G = (N_G, E_G, \ell_G)$ *and* $H = (N_H, E_H, \ell_H)$ *are graphs, their* disjoint union*, denoted* $G \uplus H$, *is the graph* (N, E, ℓ) *satisfying*

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

The sequential composition *of two graphs, denoted $G * H$, is the graph* $(N, E \cup ((N_G \times \{0\}) \times (N_H \times \{1\})), \ell)$.

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.*

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

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

$$\begin{aligned}\phi(\mathcal{S} \rightsquigarrow \mathcal{T}) &= \{G \mid \exists S \in \mathcal{S}, T \in \mathcal{T}, G \cdot\!\!\rightarrow S * T\} \\ &= \{G \mid \exists G_1, G_2, G_1 \cdot\!\!\rightarrow S, G_2 \cdot\!\!\rightarrow T, G \cdot\!\!\rightarrow G_1 * G_2\} \\ &= \{G \mid \exists G_1 \in \phi(\mathcal{S}), \exists G_2 \in \phi(\mathcal{T}), G \cdot\!\!\rightarrow G_1 * G_2\} \\ &= \phi(\phi(\mathcal{S}) \rightsquigarrow \phi(\mathcal{T}))\end{aligned}$$

□

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

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

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. 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. [16].

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: $\vartriangle, \triangledown, \triangleright$ representing conjunction, disjunction, and sequence, respectively. They satisfy the following grammar:

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

where A represents a set of atomic actions. In practice we can make the operators 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.

Definition 6 (Series-parallel). *A* series-parallel graph *over a set of possible nodes N is defined by the following grammar.*

$$G :: \; N \mid \; G \uplus G \;\mid\; G \cdot G$$

The original semantics for sequential attack trees given in [7] is the following.

$$\begin{array}{rlrl} [\![a]\!]_{\mathcal{M}} &= \{N_a\} & [\![t_1 \triangledown t_2]\!]_{\mathcal{M}} &= [\![t_1]\!]_{\mathcal{M}} \cup [\![t_2]\!]_{\mathcal{M}} \\ [\![t_1 \vartriangle t_2]\!]_{\mathcal{M}} &= [\![t_1]\!]_{\mathcal{M}} \bowtie [\![t_2]\!]_{\mathcal{M}} & [\![t_1 \triangleright t_2]\!]_{\mathcal{M}} &= [\![t_1]\!]_{\mathcal{M}} \rightsquigarrow [\![t_2]\!]_{\mathcal{M}} \end{array} \tag{$\vartriangle$}$$

The semantics given in equation ($\vartriangle$) 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. Here we give it a special symbol because it will, by analogy, play a pivotal role in what follows, as becomes manifest in Theorem 3.

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

above semantics, one involving *up-sets* and the other involving *down-sets* of graphs.[1]

The *up-set semantics* is given by the following:

$$\begin{array}{ll} [\![a]\!]_{\mathcal{F}} = \{N_a\} & [\![t_1 \triangledown t_2]\!]_{\mathcal{F}} = [\![t_1]\!]_{\mathcal{F}} \cup [\![t_2]\!]_{\mathcal{F}} \\ [\![t_1 \triangle t_2]\!]_{\mathcal{F}} = [\![t_1]\!]_{\mathcal{F}} \bowtie [\![t_2]\!]_{\mathcal{F}} & [\![t_1 \triangleright t_2]\!]_{\mathcal{F}} = \phi([\![t_1]\!]_{\mathcal{F}} \rightsquigarrow [\![t_2]\!]_{\mathcal{F}}) \end{array} \tag{2}$$

The *down-set semantics*[2] is given by the following:

$$\begin{array}{ll} [\![a]\!]_{\mathcal{I}} = \iota(\{N_a\}) & [\![t_1 \triangledown t_2]\!]_{\mathcal{I}} = [\![t_1]\!]_{\mathcal{I}} \cup [\![t_2]\!]_{\mathcal{I}} \\ [\![t_1 \triangle t_2]\!]_{\mathcal{I}} = \iota([\![t_1]\!]_{\mathcal{I}} \bowtie [\![t_2]\!]_{\mathcal{I}}) & [\![t_1 \triangleright t_2]\!]_{\mathcal{I}} = \iota([\![t_1]\!]_{\mathcal{I}} \rightsquigarrow [\![t_2]\!]_{\mathcal{I}}) \end{array} \tag{3}$$

These two semantics generate two natural preorders on attack trees. Namely,

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

The purpose of these preorders is to enable quantitative comparisons among attack trees. There are methods of assigning quantitative values to attack trees to represent various aspects such as the minimum/maximum time required to complete an attack, or the amount of resources required to maximize parallelism. Values are assigned to the atomic actions and functions are assigned to the operators. The quantitative value of a tree arises from evaluating the quantitative expression under this substitution. Such substitution schemes are called *attribute domains* [6].

As the actual quantities may vary, it is desirable to know when quantitative comparisons will be robust to changes in the underlying quantities presuming the relative order of the values assigned to the leaves remains the same. That is, if f is a function arising from an attribute domain that assigns quantities to attack trees we would like to know whether, for example, f is sound with respect to $\preceq_{\mathcal{F}}$. That is, whether $t_1 \preceq_{\mathcal{F}} t_2$ corresponds to $f(t_1) \leqslant f(t_2)$ (or perhaps $f(t_2) \leqslant f(t_1)$). We delay further discussion of attribute domains to Section 5, noting only that as long as an attribute domain is sound with respect to one of the preorders defined above, it is sufficient to work directly with the preorder.

Specialization using $[\![\cdot]\!]_{\mathcal{M}}$. The two semantics in equations (2) and (3) interleave the graph operations with the upward and downward closure operations. Our first novel insight regarding these two semantics is that they are equivalent to first applying the semantics from [7], then applying either the upward or the downward closure.

[1] In [6], the authors call these the filter and ideal semantics respectively. As per the discussion in Section 2, we avoid those terms here but keep the suggestive use of $\mathcal{F}$ and $\mathcal{I}$.

[2] This semantics differs from the corresponding ideal semantics in [6] due to our use of arbitrary injective homomorphisms instead of smoothing homomorphisms. It is interesting that the up-set semantics remains the same under this change.

Theorem 1. *For any attack tree t, $[\![t]\!]_{\mathcal{F}} = \phi([\![t]\!]_{\mathcal{M}})$.*

Proof. We proceed by induction on the structure of t. We start with the case in which the tree is an atom a. $[\![a]\!]_{\mathcal{F}} = \{N_a\}$. And $[\![a]\!]_{\mathcal{M}} = \{N_a\} = \phi(\{N_a\})$.

When $t = t_1 \triangledown t_2$ we have

$$\begin{aligned}
[\![t_1 \triangledown t_2]\!]_{\mathcal{F}} &= [\![t_1]\!]_{\mathcal{F}} \cup [\![t_2]\!]_{\mathcal{F}} \\
&= \phi([\![t_1]\!]_{\mathcal{M}}) \cup \phi([\![t_2]\!]_{\mathcal{M}}) \\
&= \phi([\![t_1]\!]_{\mathcal{M}} \cup [\![t_2]\!]_{\mathcal{M}}) \\
&= \phi([\![t_1 \triangledown t_2]\!]_{\mathcal{M}})
\end{aligned}$$

where the second-to-last equality is by Lemma 1. The other inductive cases follow analogously from Lemma 1.

The analogous result holds for the down-set semantics.

Theorem 2. *For any attack tree t, $[\![t]\!]_{\mathcal{I}} = \iota([\![t]\!]_{\mathcal{M}})$.*

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

Theorems 1 and 2 say that one can first compute $[\![t]\!]_{\mathcal{M}}$ and then compute either the upward or the downward closure. In fact, we can do a little better. By examining the homomorphisms that exist between $[\![t_1]\!]_{\mathcal{M}}$ and $[\![t_2]\!]_{\mathcal{M}}$ we can easily determine whether $t_1 \preceq_{\mathcal{F}} t_2$ or $t_1 \preceq_{\mathcal{I}} t_2$. That is, by reducing the question of (potentially infinite) set membership of $\iota([\![t_1]\!]_{\mathcal{M}}) \subseteq \iota([\![t_2]\!]_{\mathcal{M}})$ to a (finite) question of homomorphisms that exist between finite sets of graphs, we obtain a simple decision procedure. To that end, we define the following two concepts.

Definition 7 (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 $G \cdot\!\!\rightarrow H$. We say that $\mathcal{T}$* covers *$\mathcal{S}$ iff for every $G \in \mathcal{S}$ there is some $H \in \mathcal{T}$ such that $G \cdot\!\!\rightarrow H$.*

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 $\mathcal{T}$, $\phi(\mathcal{S}) \subseteq \phi(\mathcal{T})$ if and only if $\mathcal{T}$ covers $\mathcal{S}$. Similarly, $\iota(\mathcal{S}) \subseteq \iota(\mathcal{T})$ if and only if $\mathcal{T}$ supports $\mathcal{S}$.*

Proof. Suppose $\phi(\mathcal{S}) \subseteq \phi(\mathcal{T})$. We have $\mathcal{S} \subseteq \phi(\mathcal{S}) \subseteq \phi(\mathcal{T}) = \{G \mid \exists H \in \mathcal{T}, G \cdot\!\!\rightarrow H\}$. So, for every $G \in \mathcal{S}$, there is some $H \in \mathcal{T}$ such that $G \cdot\!\!\rightarrow H$. 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 $G \cdot\!\!\rightarrow H$. Now let $K \in \phi(\mathcal{S})$, so that there is some $G \in \mathcal{S}$ such that $K \cdot\!\!\rightarrow G$. But from above, there is some $H \in \mathcal{T}$ such that $G \cdot\!\!\rightarrow H$. Composing the homomorphisms we find $K \cdot\!\!\rightarrow H$. Since K was chosen abritrarily from $\phi(\mathcal{S})$ we conclude that $\phi(\mathcal{S}) \subseteq \phi(\mathcal{T})$ as required.

Now suppose that $\iota(\mathcal{S}) \subseteq \iota(\mathcal{T})$. We have $\mathcal{S} \subseteq \iota(\mathcal{S}) \subseteq \iota(\mathcal{T}) = \{H \mid \exists G \in \mathcal{T}, G \cdot\!\!\rightarrow H\}$. So for all $H \in \mathcal{S}$ there is some $G \in \mathcal{T}$ such that $G \cdot\!\!\rightarrow H$. 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 $G \cdot\!\!\rightarrow H$. Now consider $K \in \iota(\mathcal{S})$. By definition, there is some $H \in \mathcal{S}$ such that $H \cdot\!\!\rightarrow K$. But from above, there is some $G \in \mathcal{T}$ such that $G \cdot\!\!\rightarrow H$. So composing the homomorphisms, we find $G \cdot\!\!\rightarrow K$, showing that $K \in \iota(\mathcal{T})$. Since K was chosen arbitrarily from $\iota(\mathcal{S})$, we conclude $\iota(\mathcal{S}) \subseteq \iota(\mathcal{T})$ as required. □

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 upward and downward closures. Thus, we can hang our analysis on the original semantics of equation (⩤).

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

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

We would like to emphasize that the up- and down-set semantics as defined in [6] do not generate infinite sets because their upward and downward closures are taken with respect to smoothing homomorphisms (which are bijections on the nodes of the graphs). This bounds the amount of information that can be added or removed by homomorphisms, resulting in finite sets. Our decision to consider more general injective homomorphisms therefore generates a new problem which is resolved by Corollary 1. We will see additional payoffs of using the more general injective homomorphisms in later sections.

Despite the finiteness of the semantics in [6], they ultimately provide a more tractable method for deciding whether two attack trees can be ordered. They do this by developing 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 6.

We repeatedly use Theorem 3 throughout the paper to get the benefits of up- and down-set semantics without having to compute those sets directly. For structures other than attack trees, we will identify a "base" semantics analogous to equation (⩤), and then define preorders according to which sets in those semantics cover or support which others.

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:

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

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 $@_P$ which is almost purely about data flow. 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 original (sequential) attack tree semantics 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 t, this evidence-type semantics, denoted $\mathcal{E}(t, p, e)$, is sensitive to the entity 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.

We can now specify the series-parallel graph semantics for Copland phrases.

$$
\begin{aligned}
[\![\, a(\bar{v}) \,]\!]_p^e &= N_a(\bar{v}, p, e) \\
[\![\, @_q\, t \,]\!]_p^e &= \mathsf{req}(p, q) * [\![\, t \,]\!]_q^e * \mathsf{rpy}(p, q) \\
[\![\, t_1 \rightarrow t_2 \,]\!]_p^e &= [\![\, t_1 \,]\!]_p^e * [\![\, t_2 \,]\!]_p^{\mathcal{E}(t_1, p, e)} \\
[\![\, t_1 \overset{\pi}{\prec} t_2 \,]\!]_p^e &= \mathsf{split}(p, \pi) * [\![\, t_1 \,]\!]_p^{\pi_1(e)} * [\![\, t_2 \,]\!]_p^{\pi_2(e)} * \mathsf{joins}(p) \\
[\![\, t_1 \overset{\pi}{\sim} t_2 \,]\!]_p^e &= \mathsf{split}(p, \pi) * ([\![\, t_1 \,]\!]_p^{\pi_1(e)} \uplus [\![\, t_2 \,]\!]_p^{\pi_2(e)}) * \mathsf{joinp}(p)
\end{aligned}
\tag{4}
$$

There is enough detail in (4) to warrant a more detailed comparison with the attack tree semantics from (⩓). 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 $t_1 \rightarrow t_2$, t_2 is evaluated with the evidence produced by t_1. By contrast, in $t_1 \overset{\pi}{\prec} t_2$, t_2 is evaluated with $\pi_2(e)$ which is derived from the evidence type built up before t_1 and t_2 are sequenced.

The reader will also have noticed the existence of "extra" events such as $\mathsf{req}(p, q)$, $\mathsf{split}(p, \pi)$, etc. These 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 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.

Table 1. Correspondence between Copland and Attack Tree operators.

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

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

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 equations (2) and (3). But it is not immediately obvious how to interleave the upward and downward closures with the series-parallel graph operations. Using Theorem 3, we can avoid taking upward and downward closures altogether and define two preorders on Copland phrases as follows:

$$\begin{aligned} t_1 \preceq^C_{\mathcal{F}} t_2 &\text{ iff } \{[\![\, t_2 \,]\!]^e_p\} \text{ covers } \{[\![\, t_1 \,]\!]^e_p\} \\ t_1 \preceq^C_{\mathcal{I}} t_2 &\text{ iff } \{[\![\, t_2 \,]\!]^e_p\} \text{ supports } \{[\![\, t_1 \,]\!]^e_p\} \end{aligned} \tag{5}$$

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

$$\begin{aligned} t_1 \preceq^C_{\mathcal{F}} t_2 &\text{ iff } [\![\, t_1 \,]\!]^e_p \cdot\!\!\rightarrow [\![\, t_2 \,]\!]^e_p \\ t_1 \preceq^C_{\mathcal{I}} t_2 &\text{ iff } [\![\, t_2 \,]\!]^e_p \cdot\!\!\rightarrow [\![\, t_1 \,]\!]^e_p \end{aligned} \tag{6}$$

Notice that, since the Copland semantics produces a single series-parallel graph, $t_1 \preceq^C_{\mathcal{F}} t_2$ iff $t_2 \preceq^C_{\mathcal{I}} t_1$. This is not true in general for semantics that result in sets of graphs.

5 Attribute Domains

In this section we demonstrate how to leverage the connection between attack trees and Copland by considering some attribute domains for which the preorders are sound.

Definition 8 (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 the operators of an attack tree or a Copland phrase. 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.*

Attribute domains provide a way of giving quantitative values to attack trees or Copland phrases provided a base function $\nu : A \to V$ is provided. 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.

Definition 9 (Soundness). *An attribute domain D is* sound *with respect to a given preorder $\preceq$ if and only if for all t_1, t_2, ν, either*

- $t_1 \preceq t_2$ *implies* $\mathcal{V}_\nu(t_1) \leqslant \mathcal{V}_\nu(t_2)$*, or*
- $t_1 \preceq t_2$ *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 $\preceq_{\mathcal{F}}$ and $\preceq_{\mathcal{I}}$. These are presented in Table 2.

Table 2. Some sound attribute domains.

Attribute domain	preorder	Soundness Direction	Interpretation
(ℕ,min,+,max)	$\preceq_{\mathcal{F}}$	Contra-variant	Minimum experts required
(ℝ,min,max,+)	$\preceq_{\mathcal{I}}$	Contra-variant	Minimum attack time
(ℕ,max,+,max)	$\preceq_{\mathcal{I}}$	Co-variant	Guards needed to counter attack
(ℝ,max,max,+)	$\preceq_{\mathcal{F}}$	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. This correspondence is depicted in Table 3.

Table 3. Correspondence between attack tree and Copland attribute domains.

Attack Trees	Copland
($\mathbb{N}$,min,+,max)	($\mathbb{N}$,max,max,+,0)
($\mathbb{R}$,min,max,+)	($\mathbb{R}$,+,+,max,0)
($\mathbb{N}$,max,+,max)	($\mathbb{N}$,max,max,+,0)
($\mathbb{R}$,max,max,+)	($\mathbb{R}$,+,+,max,0)

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 $\mathsf{req}, \mathsf{rpy}, \mathsf{split}$, 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 $(\mathbb{R}, +, +_{s+j}, \max_{s+j}, q + p)$ 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. It is a simple exercise to verify that this attribute domain is co-variantly sound with respect to $\preceq^C_{\mathcal{F}}$ and contra-variantly sound with respect to $\preceq^C_{\mathcal{I}}$, given that row 2 of Table 3 is also sound in the same way.

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 $\preceq^C_{\mathcal{F}}$ and $\preceq^C_{\mathcal{I}}$. However, the correspondence in Table 3 only suggests two attribute domains from the four identified in [6]. 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 Cryptographic Protocols

We make a brief detour into the world of cryptographic protocols to report on a surprising connection to the specialization of attack trees. 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 conjecture below, the method for generating this preorder can be viewed as a generalized instance of the preorders defined above for attack trees and Copland phrases. Due to space limitations, we can only give a very high-level overview.

The general idea 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. 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 to some height 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 [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 $\mathbb{A} \cdot\!\!\rightarrow \mathbb{B}$. Furthermore, $\mathcal{S}_\mathbb{A}(P)$ supports the set of all realized skeletons $\mathbb{C}$ satisfying $\mathbb{A} \cdot\!\!\rightarrow \mathbb{C}$. That is, for all realized skeletons $\mathbb{C}$ such that $\mathbb{A} \cdot\!\!\rightarrow \mathbb{C}$, there is an element $\mathbb{B} \in \mathcal{S}_\mathbb{A}(P)$ such that $\mathbb{B} \cdot\!\!\rightarrow \mathbb{C}$.

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 using Theorem 3 directly on the sets $\mathcal{S}_{\mathbb{A}_1}(P_1)$ and $\mathcal{S}_{\mathbb{A}_2}(P_2)$ to generate a preorder. The translation into logic acts as a substitute. 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 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 first remark that, 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 suggests that the approach can be generalized to other structures equipped with natural homomorphisms. Additionally, the conjecture would not hold if we restricted attention to smoothing homomorphisms only. $P_1 \preceq_\Phi P_2$ 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 to use general injective homomorphisms in Section 3 that resulted in an alteration to the down-set semantics compared to [6].

A much bigger difference from the preceding formalisms is that the semantics of protocols is not tied to the protocol syntax the way it is for attack trees and Copland. Soundness of attribute domains is connected with certain algebraic aspects of the syntax of attack trees. The ways in which operators in attack trees commute and distribute determine constraints on how their associated functions must commute and distribute in attribute domains. Since the semantics of cryptographic protocols do not reflect any algebraic structure in the protocol syntax, we cannot talk about attribute domains or their soundness with respect to the protocol semantics. If anything, the translation into logic is a kind of domain that comes equipped with its own preorder. But this does not fit the mold of attribute domains as described in the previous section. Related to this

fact, is the fact that the graph-like structures are not restricted to be series-parallel. The series-parallel structure of attack tree and Copland semantics is related to the algebraic properties of their semantics. This deviation from the series-parallel paradigm, however, does not stand in the way of constructing preorders according to Theorem 3.

Note also that each preorder is "relativized" to a set of assumptions. It may be that two protocols are ordered one way relative to Φ_1 but ordered the opposite way relative to Φ_2. This presents no contradiction because each Φ_i gives rise to its own independent preorder.

Finally, notice that the conjecture only uses the notion of "covers" and not the notion of "supports". Thus, $\preceq_\Phi$ is a kind of up-set semantics. That is, it shares the same form as the $\preceq_\mathcal{F}$ order on attack trees. If we write it as $\preceq^\Phi_\mathcal{F}$, this suggests the natural alternative $\preceq^\Phi_\mathcal{I}$ defined according to a down-set semantics. That is, which $P_1 \preceq^\Phi_\mathcal{I} P_2$ 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) $\preceq^\Phi_\mathcal{I}$.

7 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.

The material in Section 6 suggests that, if we had an "adversary-enriched semantics" for Copland phrases, we could leverage Theorem 3 to develop a preorder based on it. Such a semantics would necessarily differ from the Copland semantics of equation (4) which does not account for adversary actions. Conjecture 1 gives us hope that we may diverge from a semantics with a tight connection to the syntax of a Copland phrase and still identify a preorder that might capture an intuitive notion of trust.

In fact, our prior work on layered attestations [12] established an adversary model that leads to such a semantics. 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]. Just as CPSA computes a set of support for all executions of a protocol consistent with initial assumptions, our tool chain computes something similar for Copland

phrases. We find a set of support for all executions of a Copland phrase consistent with the assumptions that some set of components are corrupt at the time they are measured, and that all corruptions go undetected.

Let us denote this set by $\mathcal{A}_H(t)$, where $\mathcal{A}$ indicates it is an adversary-enriched semantics, and H denotes the hypotheses assumed (e.g. some set of components assumed to have been corrupted). This is a set of graphs with extra structure to encode assumptions about which components are corrupt at which events. Using Theorem 3, we again define two preorders on Copland phrases. This time they are parameterized by the hypotheses H.

$$\begin{aligned} t_1 \preceq^H_{\mathcal{F}} t_2 \text{ iff } \mathcal{A}_H(t_2) \text{ covers } \mathcal{A}_H(t_1) \\ t_1 \preceq^H_{\mathcal{I}} t_2 \text{ iff } \mathcal{A}_H(t_2) \text{ supports } \mathcal{A}_H(t_1) \end{aligned} \tag{7}$$

The development of equation (7) is a largely mechanical application of Theorem 3 to generate two preorders. However, our main interest in developing preorders that capture 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 equation (7) 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.

Expressed more formally, we can define a mapping RD (for recent or deep) of Copland phrases into the 2-point lattice $\{\bot, \top\}$. $RD(t) = \top$ if the only way for an adversary to avoid detection is by employing recent or deep strategies. $RD(t) = \bot$ if there is some strategy that allows the adversary to avoid detection that is neither recent nor deep. 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 : Copland $\rightarrow \{\bot, \top\}$. At a coarse level, then, we consider a Copland phrase t to be sufficiently trustworthy relative to hypotheses H if $RD_H(t) = \top$, and untrustworthy otherwise.

We can now ask whether either of the preorders of equation (7) capture the notion of trust described above. In other words, we can ask if either of $t_1 \prec^H_{\mathcal{F}} t_2$ or $t_1 \prec^H_{\mathcal{I}} t_2$ implies the same (or possibly opposite) order on $RD_H(t_1)$ and $RD_H(t_2)$. 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 $\mathsf{m}_1(a, b)$ represents the measurement of some component b by a well-protected component a. Atomic phrase $\mathsf{m}_2(b, c)$ represents the measurement of component c by component b. Thus, a represents a "deep" component of the system. There are (at least) three natural ways to order these measurements.

$$t_1 = \mathsf{m}_1(a, b) \overset{\pi}{\sim} \mathsf{m}_2(b, c) \tag{8}$$

$$t_2 = \mathsf{m}_2(b, c) \overset{\pi}{\prec} \mathsf{m}_1(a, b) \tag{9}$$

$$t_3 = \mathsf{m}_1(a, b) \overset{\pi}{\prec} \mathsf{m}_2(b, c) \tag{10}$$

Using the tool chain we developed in [14] we can compute $\mathcal{A}_H(t_i)$ for $1 \leqslant i \leqslant 3$, where H is the hypothesis that c 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.

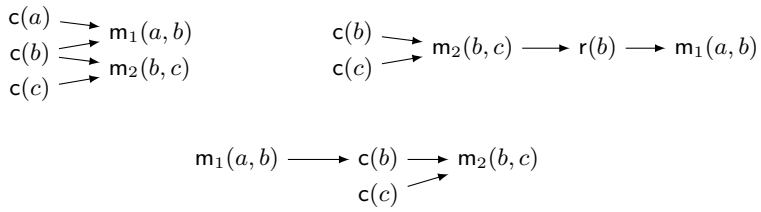

Fig. 1. The three graphs in $\mathcal{A}_H(t_1)$.

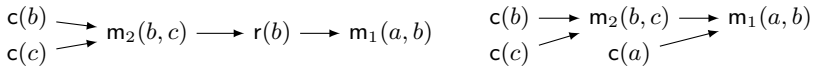

Fig. 2. The two graphs in $\mathcal{A}_H(t_2)$.

Phrases t_1 and t_2 each admit executions in which the deep component a is not corrupted, and the corruptions of b and c need not occur recently (e.g. after the start of the attestation). Thus $RD_H(t_1) = RD_H(t_2) = \bot$. In the executions admitted by t_3, there is either a deep corruption of a, or there is a recent corruption of b. Thus $RD_H(t_3) = \top$.

Fig. 3. The two graphs in $\mathcal{A}_H(t_3)$.

We can similarly determine which of the sets $\mathcal{A}_H(t_i)$ cover or support which others. It is a simple exercise to check that $\mathcal{A}_H(t_1)$ supports both $\mathcal{A}_H(t_2)$ and $\mathcal{A}_H(t_3)$, 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 $\preceq^H_{\mathcal{F}}$ and $\preceq^H_{\mathcal{I}}$ that hold are $t_2 \preceq^H_{\mathcal{I}} t_1$ and $t_3 \preceq^H_{\mathcal{I}} t_1$.

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

Theorem 4. *If* $t_1 \preceq^H_{\mathcal{I}} t_2$ *then* $t_2 \preceq^H_{RD} t_1$.

Proof. Since $t_2 \preceq^H_{RD} t_1$ holds for all values of $RD_H(t_1)$ and $RD_H(t_2)$ *except* $RD_H(t_2) = \bot$ and $RD_H(t_1) = \top$, it suffices to show that whenever $t_1 \preceq^H_{\mathcal{I}} t_2$ and $RD_H(t_1) = \top$ then $RD_H(t_2) = \top$ as well.

First note that $RD_H(t_1) = \top$ means that for every $G \in \mathcal{A}_H(t_1)$, G contains a recent corruption or a deep corruption. Also, recent and deep corruptions are both preserved under homomorphisms. Since $t_1 \preceq^H_{\mathcal{I}} t_2$, we know that for every $G_2 \in \mathcal{A}_H(t_2)$, there is some $G_1 \in \mathcal{A}_H(t_1)$ such that $G_1 \cdot\!\!\rightarrow G_2$. But since G_1 has a recent or deep corruption, this is preserved by the homomorphism to G_2. Thus every element of $\mathcal{A}_H(t_2)$ has a recent or deep corruption. So $RD_H(t_2) = \top$. □

Theorem 4 is encouraging. It shows that the generalized down-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 4 is a little unsatisfying. For one thing, the same soundness does not hold for the strict preorders. This is evident from the fact that $t_2 \prec^H_{\mathcal{I}} t_1$ but $t_1 \not\preceq^H_{RD} t_2$. 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 $t_1 \preceq^H_{\mathcal{I}} t_2$ for our purposes if we know $RD_H(t_2) = \top$ or $RD_H(t_1) = \bot$. The fact that t_2 and t_3 are $\preceq^H_{\mathcal{I}}$-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 $\preceq_{\mathcal{I}}^{H}$. This indicates that $\preceq_{\mathcal{I}}^{H}$ is, in some sense, not sensitive enough.

Theorem 4 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 $\preceq_{\mathcal{I}}^{H}$ 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 $\mathcal{A}_H(t)$ 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.

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 [8,17]. 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. Pedro Adão, Riccardo Focardi, Joshua D. Guttman, and Flaminia L. Luccio. Localizing firewall security policies. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 194–209. IEEE Computer Society, 2016.
2. Brian A. Davey and Hilary A. Priestley. *Introduction to lattices and order*. Cambridge University Press, Cambridge, 1990.
3. Joshua D. Guttman. Establishing and preserving protocol security goals. *J. Comput. Secur.*, 22(2):203–267, 2014.
4. Sarah C. Helble, Ian D. Kretz, Peter A. Loscocco, John D. Ramsdell, Paul D. Rowe, and Perry Alexander. Flexible mechanisms for remote attestation, to appear.
5. Ross Horne. The consistency and complexity of multiplicative additive system virtual. *Sci. Ann. Comput. Sci.*, 25(2):245–316, 2015.
6. Ross Horne, Sjouke Mauw, and Alwen Tiu. Semantics for specialising attack trees based on linear logic. *Fundam. Informaticae*, 153(1-2):57–86, 2017.
7. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In Hannes Federrath and Dieter Gollmann, editors, *ICT Systems Security and Privacy Protection - 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, May 26-28, 2015, Proceedings*, volume 455 of *IFIP Advances in Information and Communication Technology*, pages 339–353. Springer, 2015.
8. Gordon D. Plotkin. A powerdomain construction. *SIAM J. Comput.*, 5(3):452–487, 1976.
9. John D. Ramsdell. Deducing security goals from shape analysis sentences. *CoRR*, abs/1204.0480, 2012.
10. John D. Ramsdell, Joshua D. Guttman, Moses D. Liskov, and Paul D. Rowe. The CPSA specification: A reduction system for searching for shapes in cryptographic protocols, 2012.
11. John D. Ramsdell, Paul D. Rowe, Perry Alexander, Sarah Helble, Peter Loscocco, J. Aaron Pendergrass, and Adam Petz. Orchestrating layered attestations. In Flemming Nielson and David Sands, editors, *Principles of Security and Trust - 8th International Conference, POST 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11426 of *Lecture Notes in Computer Science*, pages 197–221. Springer, 2019.
12. Paul D. Rowe. Confining adversary actions via measurement. *Third International Workshop on Graphical Models for Security*, pages 150–166, 2016.
13. Paul D. Rowe, Joshua D. Guttman, and Moses D. Liskov. Measuring protocol strength with security goals. *Int. J. Inf. Sec.*, 15(6):575–596, 2016.
14. Paul D. Rowe, John D. Ramsdell, and Ian D. Kretz. Automated trust analysis for layered attestations, in preparation.
15. Bruce Schneier. Attack trees, 1999.
16. Wojciech Wideł, Maxime Audinot, Barbara Fila, and Sophie Pinchinat. Beyond 2014: Formal methods for attack tree-based security modeling. *ACM Comput. Surv.*, 52(4):75:1–75:36, 2019.
17. Glynn Winskel. On powerdomains and modality. *Theor. Comput. Sci.*, 36:127–137, 1985.