Hyperproperties*

Michael R. Clarkson   Fred B. Schneider
{clarkson,fbs}@cs.cornell.edu
Department of Computer Science
Cornell University

October 16, 2009

Abstract

Trace properties, which have long been used for reasoning about systems, are sets of execution traces. Hyperproperties, introduced here, are sets of trace properties. Hyperproperties can express security policies, such as secure information flow and service level agreements, that trace properties cannot. Safety and liveness are generalized to hyperproperties, and every hyperproperty is shown to be the intersection of a safety hyperproperty and a liveness hyperproperty. A verification technique for safety hyperproperties is given and is shown to generalize prior techniques for verifying secure information flow. Refinement is shown to be applicable with safety hyperproperties. A topological characterization of hyperproperties is given.

1 Introduction

Important security policies cannot be expressed as properties of individual execution traces of a system [2, 22, 42, 52, 60, 62, 64]. For example, *noninterference* [23] is a confidentiality policy that stipulates commands executed on behalf of users holding high clearances have no effect on system behavior observed by users holding low clearances. It is not a property of individual traces, because whether a trace is allowed by the policy depends on whether another trace (obtained by deleting command executions by high users) is also allowed. For another example, stipulating a bound on mean response time over all executions is an availability policy that cannot be specified as a property of individual traces, because the acceptability of delays in a trace depends on the magnitude of delays in all other traces. However, both example policies are properties of systems, because a system (viewed as a whole, not as individual executions) either does or does not satisfy each policy.

*Supported in part by AFOSR grant F9550-06-0019, National Science Foundation Grants 0430161 and CCF-0424422 (TRUST), an Intel Foundation Ph.D. Fellowship, and a gift from Microsoft Corporation.

A *property* either holds or does not hold (i.e., is a Boolean function) of an object, and the *extension* of a property is the set of objects for which the property holds. The extension of a property of individual traces—that is, a set of traces—sometimes is termed "property," too [4, 35]. But for clarity, *trace property* here denotes a set of traces.

The theory of trace properties is well understood [36, 37, 54]. Every trace property is the intersection of a safety property and a liveness property, where

- a *safety property* is a trace property that proscribes "bad things" and can be proved using an invariance argument, and
- a *liveness property* is a trace property that prescribes "good things" and can be proved using a well-foundedness argument.[1]

This classification forms an intuitively appealing basis from which all trace properties can be constructed. Moreover, safety and liveness properties are affiliated with specific verification methods.

An analogous theory for security policies would be appealing. The fact that security policies, like trace properties, proscribe and prescribe behaviors of systems suggested that such a theory might exist. This paper develops that theory by formalizing security policies as properties of systems, or *system properties*.[2] If systems are modeled as sets of execution traces [35], then the extension of a system property is a set of sets of traces or, equivalently, a set of trace properties. We name this type of set a *hyperproperty*.

Every property of system behavior (for systems modeled as trace sets) can be specified as a hyperproperty, by definition. Thus, hyperproperties can describe trace properties and moreover can describe security policies, such as noninterference and mean response time, that trace properties cannot. Deterministic, nondeterministic, probabilistic, and transition-system models all can be encoded as trace sets and handled using hyperproperties.

This paper shows that results similar to those from the theory of trace properties hold for hyperproperties:

- Every hyperproperty is the intersection of a safety hyperproperty and a liveness hyperproperty. (Henceforth, we shorten these terms to *hypersafety* and *hyperliveness*.) Hypersafety and hyperliveness thus form a basis from which all hyperproperties can be constructed.
- Hyperproperties from a class that we introduce, called k*-safety*, can be verified by using invariance arguments. Our verification methodology generalizes prior work on using invariance arguments to verify information-flow policies [8, 60].

However, we have not obtained complete verification methods for hypersafety or for hyperliveness.

[1]Lamport [33] gave the first informal definitions of safety and liveness properties, appropriating the names from Petri net theory, and he also gave the first formal definition of safety [35]. Alpern and Schneider [4] gave the first formal definition of liveness and the proof that all trace properties are the intersection of safety and liveness properties; they later established the correspondence of safety to invariance and of liveness to well-foundedness [5].

[2]McLean [42] gave the first formalization of security policies as properties of trace sets.

The theory of hyperproperties also sheds light on the problematic status of refinement for security policies. Refinement never invalidates a trace property but can invalidate a hyperproperty:

> Consider a system π that nondeterministically chooses to output 0, 1, or the value of a secret bit h. System π satisfies the security policy "The possible output values are independent of the values of secrets." But one refinement of π is the system that always outputs h, and this system does not satisfy the security policy.

We characterize in this paper the entire set of hyperproperties for which refinement is valid; this set includes the safety hyperproperties.

Safety and liveness not only form a basis for trace properties and hyperproperties, but they also have a surprisingly deep mathematical characterization in terms of topology. In the *Plotkin* topology on trace properties, safety and liveness are known to correspond to *closed* and *dense* sets, respectively [4]. We generalize this topological characterization to hyperproperties by showing that hypersafety and hyperliveness also correspond to closed and dense sets in a new topology, which turns out to be equivalent to the *lower Vietoris* construction applied to the Plotkin topology [57]. This correspondence could be used to bring results from topology to bear on hyperproperties.

We proceed as follows. Hyperproperties, hypersafety, k-safety, and hyperliveness are defined and explored in sections 2, 3, 4, and 5, respectively. Section 6 gives a topological account of hyperproperties. Section 7 presents the hyperproperty intersection theorem and discusses hyperproperties of system representations other than trace sets (relational systems, labeled transition systems, state machines, and probabilistic systems). Section 8 concludes. Appendix A gives a guide to our notation, appendix B presents formal details of our longer examples of hyperproperties, appendix C states formal results about system representations, and all proofs appear in appendix D.

This paper revises and expands a CSF'08 paper [15], adding (i) new results about system representations, and (ii) proofs, which were absent from the earlier paper. Several of the proofs have been verified [12] using the Isabelle/HOL proof assistant [46].

2 Hyperproperties

We model system execution with traces, where a *trace* is a sequence of states; by employing rich enough notions of state, this model can encode other representations of execution.[3] For example, section 7 discusses how to model a labeled transition system as a set of traces by including transition labels in states, thereby preserving information about the nondeterministic branching structure of the system. Section 7 also uses this encoding to model state machines and probabilistic systems.

The structure of a state is not important in the following definitions, so we leave set Σ of states abstract. However, the structure of a state is important for real examples, and we introduce predicates and functions, on states and on traces, as needed to model events, timing, probability, etc.

[3]We have not investigated analogues to hyperproperties for representations of system execution that cannot be encoded as trace sets.

Traces may be finite or infinite sequences, which we categorize into sets:

$$\begin{aligned}\Psi_{\mathsf{fin}} &\triangleq \Sigma^*,\\ \Psi_{\mathsf{inf}} &\triangleq \Sigma^\omega,\\ \Psi &\triangleq \Psi_{\mathsf{fin}} \cup \Psi_{\mathsf{inf}},\end{aligned}$$

where Σ^* denotes the set of all finite sequences over Σ, and Σ^ω denotes the set of all infinite sequences over Σ. For trace $t = s_0 s_1 \ldots$ and index $i \in \mathbb{N}$, we define the following indexing notation:

$$\begin{aligned}t[i] &\triangleq s_i,\\ t[..i] &\triangleq s_0 s_1 \ldots s_i,\\ t[i..] &\triangleq s_i s_{i+1} \ldots\end{aligned}$$

We denote concatenation of finite trace t and (finite or infinite) trace t' as tt', and we denote the empty trace as ϵ.

A *system* is modeled by a non-empty set of infinite traces, called its *executions*. If an execution terminates (and thus could be represented by a finite trace), we represent it as an infinite trace by infinitely stuttering the final state in the finite trace.

2.1 Trace Properties

A *trace property* is a set of infinite traces [4, 35]. The set of all trace properties is

$$\mathsf{Prop} \triangleq \mathcal{P}(\Psi_{\mathsf{inf}}),$$

where $\mathcal{P}$ denotes powerset. A set T of traces satisfies a trace property P, denoted $T \models P$, iff all the traces of T are in P:

$$T \models P \triangleq T \subseteq P.$$

Some security policies are expressible as trace properties. For example, consider the policy "The system may not write to the network after reading from a file." Formally, this is the set of traces

$$\begin{aligned}\mathit{NRW} \triangleq \{t \in \Psi_{\mathsf{inf}} \mid \neg(\exists\, i, j \in \mathbb{N} : i < j \;\wedge\; &\mathit{isFileRead}(t[i])\\ &\wedge\; \mathit{isNetworkWrite}(t[j]))\},\end{aligned} \quad (2.1)$$

where $\mathit{isFileRead}$ and $\mathit{isNetworkWrite}$ are state predicates.

Similarly, *access control* is a trace property requiring every operation to be consistent with its requestor's rights:

$$\begin{aligned}AC \triangleq \{t \in \Psi_{\mathsf{inf}} \mid (\forall\, i \in \mathbb{N} : &\mathit{rightsReq}(t[i])\\ &\subseteq \mathit{acm}(t[i-1])[\mathit{subj}(t[i]), \mathit{obj}(t[i])])\}.\end{aligned} \quad (2.2)$$

Function $\mathit{acm}(s)$ yields the access control matrix in state s. Function $\mathit{subj}(s)$ yields the subject who requested the operation that led to state s, function $\mathit{obj}(s)$ yields the object involved in that operation, and function $\mathit{rightsReq}(s)$ yields the rights required for the operation to be allowed.

As another example, *guaranteed service* is a trace property requiring that every request for service is eventually satisfied:

$$\begin{aligned} GS \;\triangleq\; & \{t \in \Psi_{\mathsf{inf}} \mid (\forall\, i \in \mathbb{N} : \mathit{isReq}(t[i]) \\ & \qquad \Longrightarrow (\exists\, j > i : \mathit{isRespToReq}(t[j], t[i])))\}. \end{aligned} \tag{2.3}$$

Predicate $\mathit{isReq}(s)$ identifies whether a request is initiated in state s, and predicate $\mathit{isRespToReq}(s', s)$ identifies whether state s' completes the response to the request initiated in state s.

2.2 Hyperproperties

A *hyperproperty* is a set of sets of infinite traces, or equivalently a set of trace properties. The set of all hyperproperties is

$$\begin{aligned} \mathbf{\mathsf{HP}} &\triangleq \mathcal{P}(\mathcal{P}(\Psi_{\mathsf{inf}})) \\ &= \mathcal{P}(\mathsf{Prop}). \end{aligned}$$

The interpretation of a hyperproperty as a security policy is that the hyperproperty is the set of systems allowed by that policy.[4] Each trace property in a hyperproperty is an allowed system, specifying exactly which executions must be possible for that system. Thus a set T of traces satisfies hyperproperty $\boldsymbol{H}$, denoted $T \models \boldsymbol{H}$, iff T is in $\boldsymbol{H}$:

$$T \models \boldsymbol{H} \;\triangleq\; T \in \boldsymbol{H}.$$

Note the use of bold face to denote hyperproperties (e.g., $\boldsymbol{H}$) and sans serif to denote sets of trace properties (e.g., Prop). Although a hyperproperty and a set of trace properties are mathematically the same kind of object (a set of sets of traces), they are used differently in formulas, hence the different typography. Sets of hyperproperties are simultaneously bold face and sans serif (e.g., $\mathbf{\mathsf{HP}}$). See Appendix A for a guide to other typographical conventions and notation.

Given a trace property P, there is a unique hyperproperty denoted $[P]$ that expresses the same policy as P. We call this hyperproperty the *lift* of P. For P and $[P]$ to express the same policy, they must be satisfied by the same sets of traces. Thus we can derive a definition of $[P]$:

$$\begin{aligned} & (\forall T \in \mathsf{Prop} : T \models P \iff T \models [P]) \\ = \;& (\forall T \in \mathsf{Prop} : T \subseteq P \iff T \in [P]) \\ = \;& [P] = \{T \in \mathsf{Prop} \mid T \subseteq P\} \\ = \;& [P] = \mathcal{P}(P). \end{aligned}$$

Consequently, the lift of P is the powerset of P:

$$[P] \;\triangleq\; \mathcal{P}(P).$$

[4]The hyperproperty might also contain the empty set of traces, although this set does not correspond to a system.

2.3 Hyperproperties in Action

Trace properties are satisfied by traces, whereas hyperproperties are satisfied by sets of traces. This additional level of sets means that hyperproperties can be more expressive than trace properties. We explore this added expressivity with some examples.

Secure information flow. Information-flow security policies express restrictions on what information may be learned by users of a system. Users interact with systems by providing inputs and observing outputs. To model this interaction, define $ev(s)$ as the input or output event, if any, that occurs when a system transitions to state s. Assume that at most one event, input or output, can occur at each transition. For a trace t, extend this notation to $ev(t)$, denoting the sequence of events resulting from application of $ev(\cdot)$ to each state in trace t.[5] Further assume that each user of a system is cleared either at confidentiality level L, representing *low* (public) information, or H, representing *high* (secret) information, and that each event is labeled with one of these confidentiality levels. Define $ev_L(t)$ to be the subsequence of low input and output events contained within $ev(t)$, and $ev_{Hin}(t)$ to be the subsequence of high input events contained within $ev(t)$.

Noninterference, as defined by Goguen and Meseguer [23], requires that commands issued by users holding high clearances be removable without affecting observations of users holding low clearances. Treating commands as inputs and observations as outputs, we model this security policy as a hyperproperty requiring a system to contain, for any trace t, a corresponding trace t' with no high inputs yet with the same low events as t:

$$\begin{aligned}\textbf{\textit{GMNI}} \triangleq \{T \in \mathsf{Prop} \mid\ & T \in \textbf{\textit{SM}} \\ & \wedge\ (\forall\, t \in T : (\exists\, t' \in T : ev_{Hin}(t') = \epsilon \\ & \qquad \wedge\ ev_L(t) = ev_L(t')))\}. \end{aligned} \tag{2.4}$$

Conjunct $T \in$ ***SM*** expresses the requirement, made by Goguen and Meseguer's formalization, that systems are deterministic state machines; section 7.2.3 defines ***SM*** formally. ***GMNI*** is not a trace property, as argued in section 1, because trace t is allowed only if corresponding trace t' is also allowed.

Generalized noninterference [40] extends Goguen and Meseguer's definition of noninterference to handle nondeterministic systems, which are the systems modeled by Prop. McLean [42] reformulates generalized noninterference as a policy requiring a system to contain, for any traces t_1 and t_2, an interleaved trace t_3 whose high inputs are the same as t_1 and whose low events are the same as t_2. This is a hyperproperty:

$$\begin{aligned}\textbf{\textit{GNI}} \triangleq \{T \in \mathsf{Prop} \mid\ & (\forall\, t_1, t_2 \in T : (\exists\, t_3 \in T : \\ & ev_{Hin}(t_3) = ev_{Hin}(t_1)\ \wedge\ ev_L(t_3) = ev_L(t_2)))\}. \end{aligned} \tag{2.5}$$

GNI is not a trace property because the presence of any two traces t_1 and t_2 in a system necessitates the presence of a third trace t_3.

[5]Depending on the nature of events in the particular system that is being modeled, it might be appropriate for $ev(t)$ to eliminate stuttering of events.

Observational determinism [41, 51] requires a system to appear deterministic to a low user. Zdancewic and Myers's [65] definition of observational determinism can be formulated as a hyperproperty:

$$\boldsymbol{OD} \triangleq \{T \in \mathsf{Prop} \mid (\forall t, t' \in T : t[0] =_L t'[0] \implies t \approx_L t')\}. \tag{2.6}$$

State equivalence relation $s =_L s'$ holds whenever states s and s' are indistinguishable to a low user, and trace equivalence relation $t \approx_L t'$ holds whenever traces t and t' are indistinguishable to a low user. Zdancewic and Myers define trace equivalence in terms of state equivalence, requiring the sequence of states in each trace to be equivalent up to both stuttering and prefix; equivalence up to prefix makes their definition *termination insensitive*—that is, systems are allowed to leak information via termination channels.[6] ***OD*** is not a trace property because whether some trace is allowed in a system depends on all the other traces of the system.

Bisimulation-based definitions of information-flow security policies can also be formulated as hyperproperties,[7] which we demonstrate with Focardi and Gorrieri's [22] bisimulation nondeducibility on compositions (BNDC) in section 7.2.2, and with Boudol and Castellani's [11] definition of noninterference in appendix B.

All information-flow security policies we investigated turned out to be hyperproperties, not trace properties. This is suggestive, but any stronger statement about the connection between information flow and hyperproperties would require a formal definition of information-flow policies, and none is universally accepted. Nonetheless, we believe that information flow is intrinsically tied to correlations between (not within) executions. And hyperproperties are sufficiently expressive to formulate such correlations, whereas trace properties are not.

Service level agreements. A *service level agreement* (SLA) specifies acceptable performance of a system. Such specifications commonly use statistics such as

- *mean response time*, the mean time that elapses between a request and a response;
- *time service factor*, the percentage of requests that are serviced within a specified time; and
- *percentage uptime*, the percentage of time during which the system is available to accept and service requests.

These statistics can be used to define policies with respect to individual executions of a system or across all executions of a system. In the former case, the SLA would be a trace property. For example, the policy "The mean response time *in each execution*

[6]Zdancewic and Myers also require systems to be race free, hence they weaken trace equivalence to hold for each memory location in a state in isolation, not over all memory locations simultaneously. We omit this requirement for simplicity.

[7]Since hyperproperties are trace-based, this might at first seem to contradict results, such as Focardi and Gorrieri's [22], stating that bisimulation-based definitions are more expressive than trace-based definitions. However, by employing a richer notion of state [54, §1.3] in traces than Focardi and Gorrieri, hyperproperties are able to express bisimulations.

is less than 1 second" might not be satisfied by a system if there are executions in which some response times are much greater than 1 second. Yet if these executions are rare, then the system might still satisfy the policy "The mean response time *over all executions* is less than 1 second." This latter SLA is not a trace property, but it is a hyperproperty:

$$\boldsymbol{RT} \;\triangleq\; \{T \in \mathsf{Prop} \mid \mathit{mean}\left(\bigcup_{t \in T} \mathit{respTimes}(t)\right) \le 1\}. \tag{2.7}$$

Function $\mathit{mean}(X)$ denotes the mean[8] of a set X of real numbers, and $\mathit{respTimes}(t)$ denotes the set of response times (in seconds) from request/response events in trace t. Policies derived from the other SLA statistics above can similarly be expressed as hyperproperties.

2.4 Beyond Hyperproperties?

Hyperproperties are able to express security policies that trace properties cannot. So it is natural to ask whether there are security policies that hyperproperties cannot express. In section 1, we equated security policies with system properties, and we chose to model systems as trace sets. Every property of trace sets is a hyperproperty, so by definition hyperproperties are expressively complete for our formulations of "system" and "security policy." To find security policies that hyperproperties cannot express (if any exist), we would need to examine alternative notions of systems and security policies. Section 7 discusses alternative formulations of systems, but all the formulations considered there turn out to have encodings as trace sets—thus hyperproperties are complete for those formulations. We do not know whether other formulations exist that do not have such encodings.

One way to generalize the notion of a security policy is to consider policies on sets of systems—for example, *diversity* [50], which requires the systems all to implement the same functionality but to differ in their implementation details. Any such policy, however, could be modeled as a hyperproperty on a single system that is a product[9] of all the systems in the set. So hyperproperties again seem to be sufficient.

2.5 Logic and Hyperproperties

We have not given a logic in which hyperproperties may be expressed. The examples in this paper require only second-order logic. Although higher-order logic might also be useful to express hyperproperties, higher-order logic is reducible to second-order logic [56, §6.2]. So we believe that second-order logic is sufficient to express all hyperproperties. But we do not know whether the full power of second-order logic is

[8]Since X might have infinite cardinality, ***RT*** requires a definition of the mean of an infinite set (and, for some sets, this mean does not exist). We omit formalizing such a definition here; one possibility is to use the *Cesàro mean* [27].

[9]The *product* of systems T_1 and T_2 can be defined as system $\{(t_1[0], t_2[0])(t_1[1], t_1[2]) \ldots \mid t_1 \in T_1 \wedge t_2 \in T_2\}$, comprising traces over pairs of states. Generalizing, the product of a set of n systems comprises traces over n-tuples of states.

necessary to express hyperproperties of interest. This has ramifications for verification of hyperproperties, because although full second-order logic cannot be effectively and completely axiomatized, fragments of it can be [9, §2.3].[10]

2.6 Refinement and Hyperproperties

Programmers use *stepwise refinement* [1, 7, 18, 20, 36, 63] to develop, in a series of steps, a program that implements a specification. The programmer starts from the specification. Each successive step creates a more concrete specification, ultimately culminating in a specification sufficiently concrete that a computer can execute it. To prove that the final concrete specification correctly implements the original specification, the programmer argues at each step that the new concrete specification *refines* the previous specification. Specification S_1 refines specification S_2, denoted $S_1 \text{ REF } S_2$, iff every behavior permitted by S_1 is also permitted by S_2—that is, the set of behaviors of S_1 is a subset of the set of behaviors of S_2.

Specifications might describe behaviors at different levels of abstraction. For example, a specification might describe behaviors of a queue, but a refinement of that specification might use an array to implement this behavior. Or a specification might describe behaviors using critical sections, but a refinement might implement critical sections with semaphores. So programmers need techniques to relate the behaviors described by specifications. *Abstraction functions* [28, 29] and *refinement mappings* [1] have been developed for this purpose; both interpret concrete behaviors as abstract behaviors.

Generalizing from these two techniques, let an *interpretation function* be a function of type $\Psi \rightarrow \Psi$. Let IF be any class of interpretation functions that (like abstraction functions and refinement mappings) is closed under composition and contains the identity function id.[11] An interpretation function α can be lifted to $\mathsf{Prop} \rightarrow \mathsf{Prop}$ by applying α to each trace in a set:

$$\alpha(T) \triangleq \{\alpha(t) \mid t \in T\}.$$

System S *α-satisfies* trace property P, denoted $S \models_\alpha P$, iff $\alpha(S) \models P$. Notation $S \models P$, as we have used it so far, thus means that $S \models_{id} P$.

Trace property P_1 refines P_2 under interpretation α, denoted $P_1 \text{ REF}_\alpha P_2$, iff $\alpha(P_1) \subseteq P_2$. So for trace properties, satisfaction is the same relation as refinement, and subset implies refinement—that is, if C is a subset of A, then C refines A (under interpretation id). This implication is desirable, because it permits refinements that resolve non-determinism by removing traces from a system. But it is well known that

[10]It is natural to ask whether we could further reduce second-order logic to first-order. Such a reduction is possible, but only with the Henkin, rather than standard, semantics of second-order logic [9, §4.2]. We do not know which of these semantics should be preferred for hyperproperties. However, there are trace properties, and thus hyperproperties, that we conjecture cannot be expressed in first-order logic—for example, the trace property containing the single trace $pqppqqpppqqq\ldots$, where p and q are states. This suggests that the standard semantics is appropriate.

[11]Abstraction functions must also preserve data type operations, and refinement mappings must preserve externally visible components up to stuttering. But these restrictions are not relevant to our discussion.

this kind of refinement does not generally work for security policies.[12] For example, recall system π (section 1), which nondeterministically chooses to output 0, 1, or the value of a secret bit h. System π satisfies the specification "The possible output values are independent of the values of secrets," which can be formulated as a hyperproperty. But consider a system π' that always outputs h. System π' does not satisfy the specification and therefore cannot refine π, yet $\pi' \subseteq \pi$. So subset does not imply refinement for hyperproperties as it does for trace properties.

Hyperproperty $\boldsymbol{H_1}$ refines $\boldsymbol{H_2}$ under interpretation α, denoted $\boldsymbol{H_1}$ $\textsc{href}_\alpha$ $\boldsymbol{H_2}$, iff $\alpha(\boldsymbol{H_1}) \subseteq \boldsymbol{H_2}$, where $\alpha(\boldsymbol{H})$ is defined as $\{\alpha(T) \mid T \in \boldsymbol{H}\}$. A natural relationship that we would expect to hold is

$$(\forall S \in \mathsf{Prop}, \boldsymbol{H} \in \mathsf{HP} : S \models \boldsymbol{H} \iff [S] \ \textsc{href}_{id} \ \boldsymbol{H}), \tag{2.8}$$

because satisfaction and refinement intuitively should agree (as they did for trace properties). Straightforward application of definitions shows that (2.8) holds iff $\boldsymbol{H}$ is subset closed.

Thus, perhaps unsurprisingly, the set of hyperproperties with which refinement works is the set SSC of subset-closed hyperproperties:

$$\begin{aligned}\mathsf{SSC} \ \triangleq \ \{\boldsymbol{H} \in \mathsf{HP} \mid (\forall T \in \mathsf{Prop} : T \in \boldsymbol{H} \\ \implies (\forall T' \in \mathsf{Prop} : T' \subseteq T \implies T' \in \boldsymbol{H}))\}.\end{aligned}$$

The lifted trace properties are, of course, members of SSC. But SSC contains more than just the lifted trace properties. For example, observational determinism $\boldsymbol{OD}$ (2.6) is subset closed and therefore a member of SSC, but $\boldsymbol{OD}$ is not a lifted trace property.

3 Hypersafety

According to Alpern and Schneider [4], the "bad thing" in a safety property must be both

- *finitely observable*, meaning its occurrence can be detected in finite time, and
- *irremediable*, so its occurrence can never be remediated by future events.

No-read-then-write NRW (2.1) and access control AC (2.2) are both safety. The bad thing for NRW is a finite trace in which a network write occurs after a file read. This bad thing is finitely observable, because the write can be detected in some finite prefix of the trace, and irremediable, because the network write can never be undone. For AC, the bad thing is similarly a finite trace in which an operation is performed without appropriate rights.

For trace properties, a bad thing is a finite trace that cannot be a prefix of any execution satisfying the safety property. A finite trace t is a *prefix* of a (finite or infinite) trace t', denoted $t \leq t'$, iff $t' = tt''$ for some $t'' \in \Psi$.

[12]Previous work has identified refinement techniques that are valid for use with certain information-flow security policies [10, 39, 42].

Safety property. A trace property S is a *safety property* [4] iff

$$(\forall\, t \in \Psi_{\mathsf{inf}} : t \notin S \implies (\exists\, m \in \Psi_{\mathsf{fin}} : m \leq t \wedge \\ (\forall\, t' \in \Psi_{\mathsf{inf}} : m \leq t' \implies t' \notin S))).$$

Define SP to be the set of all safety properties; note that SP is itself a hyperproperty.

We generalize safety to hypersafety by generalizing the bad thing from a finite trace to a finite[13] set of finite traces. Define Obs to be the set of such *observations*:

$$\mathsf{Obs} \triangleq \mathcal{P}^{fin}(\Psi_{\mathsf{fin}}),$$

where $\mathcal{P}^{fin}(X)$ denotes the set of all finite subsets of set X. Prefix $\leq$ on sets of traces is defined as follows:[14]

$$T \leq T' \triangleq (\forall\, t \in T : (\exists\, t' \in T' : t \leq t')).$$

Note that this definition allows T' to contain traces that have no prefix in T.

Safety hyperproperty. A hyperproperty $\boldsymbol{S}$ is a *safety hyperproperty* (is *hypersafety*) iff

$$(\forall\, T \in \mathsf{Prop} : T \notin \boldsymbol{S} \implies (\exists\, M \in \mathsf{Obs} : M \leq T \\ \wedge (\forall\, T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \boldsymbol{S}))).$$

The definition of hypersafety parallels the definition of safety, but the domains involved now include an extra level of sets. Define $\mathbf{SHP}$ to be the set of all safety hyperproperties.

Some consequences of the definition of hypersafety are:

- Observational determinism ***OD*** (2.6) is hypersafety. The bad thing is a pair of traces that are not low-equivalent despite having low-equivalent initial states.
- Safety properties lift to safety hyperproperties.

  Proposition 1. $(\forall\, S \in \mathsf{Prop} : S \in \mathsf{SP} \iff [S] \in \mathbf{SHP})$.
- Set SP of all safety properties is not a safety hyperproperty: there is no bad thing that prevents an arbitrary trace property from being extended to a safety property.

Refinement of hypersafety. Stepwise refinement works with all safety hyperproperties, because safety hyperproperties are subset closed (cf. section 2.6), as stated by the following theorem.

Theorem 1. $\mathsf{SHP} \subset \mathsf{SSC}$.

[13] Infinite sets might seem to be an attractive alternative, and many of the results in the rest of this paper would still hold. However, the topological characterization given in section 6 (specifically, Propositions 4 and 5) would be sacrificed.

[14] Other definitions of trace set prefix are possible, but inconsistent with our notion of observation. We discuss this in section 6.

A consequence of Theorem 1 is that any hyperproperty that is not subset closed cannot be hypersafety. For example, generalized noninterference ***GNI*** (2.5) is not subset closed: a system containing traces t_1 and t_2 and interleaved trace t_3 might satisfy ***GNI***, but the subset containing only t_1 and t_2 would not satisfy ***GNI***. Thus ***GNI*** cannot be hypersafety.

4 Beyond 2-Safety

Safety properties enjoy a relatively complete verification methodology based on invariance arguments [5]. Although we have not obtained such a methodology for hypersafety, we can use invariance arguments to verify a class of safety hyperproperties by generalizing recent work on verification of secure information flow.

Recall that secure information flow is a hyperproperty but not a trace property. Recent work gives system transformations that reduce verifying secure information flow[15] to verifying a safety property of some transformed system: Pottier and Simonet [49] develop a type system for verifying secure information flow based on simultaneous reasoning about two executions of a program. Darvas et al. [19] show that secure information flow can be expressed in dynamic logic. Barthe et al. [8] give an equivalent formulation for Hoare logic and temporal logic, based on a self-composition construction.

Define the *sequential self-composition of* P as the program $P; P'$, where P' denotes program P, but with every variable renamed to a fresh, primed variable—for example, variable x is renamed to x'. One way to verify that P exhibits secure information flow is to establish the following trace property of transformed program $P; P'$:

> If for every low variable l, before execution $l = l'$ holds, then when execution terminates $l = l'$ still holds, no matter what the values of high variables were.

Barthe et al. generalize the self-composition operator from sequential composition to any operator that satisfies certain conditions, and they note that parallel composition satisfies these conditions. They also relax the equality constraints in the above trace property to partial equivalence relations. Terauchi and Aiken [60] further generalize the applicability of self-composition by showing that it can be used to verify any *2-safety* property, which they define informally as a "property that can be refuted by observing two finite traces."

Using hyperproperties, we can show that the above results are special cases of a more general theorem. Define a k-safety hyperproperty as a safety hyperproperty in which the bad thing never involves more than k traces:

k*-safety hyperproperty.** A hyperproperty ***S is a *k-safety hyperproperty* (is k-

[15] These reductions are possible because the particular formulations of secure information flow used in each work are actually hypersafety. A formulation that is hyperliveness—which would include all possibilistic information-flow policies, as discussed in section 5—would not be amenable to these reductions.

safety) iff

$$(\forall\, T \in \mathsf{Prop} : T \notin \boldsymbol{S} \implies (\exists\, M \in \mathsf{Obs} : M \leq T \wedge |M| \leq k \\ \wedge\ (\forall\, T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \boldsymbol{S}))).$$

This is just the definition of hypersafety with an added conjunct "$|M| \leq k$". For a particular k, define $\mathsf{KSHP}(k)$ to be the set of all k-safety hyperproperties.

As an example of a k-safety hyperproperty for any k, consider a system that stores a secret by splitting it into k shares. Suppose that an action of the system is to output a share. Then a hyperproperty of interest might be that the system cannot, across all of its executions, output all k shares (thereby outputting sufficient information for the secret to be reconstructed). We denote this k-safety hyperproperty as $\boldsymbol{SecS}_k$.

The 1-safety hyperproperties are the lifted safety properties—that is,

$$\mathsf{KSHP}(1) \ = \ \{[S] \mid S \in \mathsf{SP}\}$$

—since the bad thing for a safety property is a single trace. Thus "1-safety" and "safety" are synonymous.

The Terauchi and Aiken definition of 2-safety properties is limited to deterministic programs that are expressed in a relational model of execution (which we address further in section 7.2.1), and it ignores nonterminating traces. So their 2-safety properties are a strict subset of the 2-safety hyperproperties, $\mathsf{KSHP}(2)$. For example, observational determinism ***OD*** (2.6) is not a 2-safety property, but it is a 2-safety hyperproperty.

Define the *parallel self-composition of system* S as the product system $S \times S$ consisting of traces over $\Sigma \times \Sigma$:

$$S \times S \ \triangleq \ \{(t[0], t'[0])(t[1], t'[1]) \cdots \mid t \in S \wedge t' \in S\}.$$

Define the *k-product* of S, denoted S^k, to be the k-fold parallel self-composition of S, comprising traces over Σ^k. Self-composition $S \times S$ is equivalent to 2-product S^2.

Previous work has shown how to reduce a particular formulation of noninterference of S to a related safety property of S^2 [8], and how to reduce any 2-safety hyperproperty of system S to a related safety property of $S; S'$ [60]. The following theorem generalizes those results. Let Sys be the set of all systems. For any system S, any k-safety hyperproperty $\boldsymbol{K}$ of S can be reduced to a safety property K of S^k, and the proof of the theorem (in appendix D) shows how to construct K from $\boldsymbol{K}$:

Theorem 2. $(\forall\, S \in \mathsf{Sys}, \boldsymbol{K} \in \mathsf{KSHP}(k) : (\exists\, K \in \mathsf{SP} : S \models \boldsymbol{K} \iff S^k \models K))$.

Theorem 2 provides a verification technique for k-safety: reduce a k-safety hyperproperty to a safety property, then verify that the safety property is satisfied by S^k using an invariance argument. Since invariance arguments are relatively complete for safety properties [5], this methodology is relatively complete for k-safety.

However, Theorem 2 does not provide the relatively complete verification procedure we seek for hypersafety, because there are safety hyperproperties that are not k-safety for any k. For example, consider the hyperproperty "for any k, a system cannot output all k shares of a secret from a k-secret sharing":

$$\textbf{\textit{SecS}} \triangleq \bigcup_k \textbf{\textit{SecS}}_k. \tag{4.1}$$

SecS is not k-safety for any k. Yet it is hypersafety, since any trace property not contained in it violates some ***SecS***$_k$.

5 Hyperliveness

Alpern and Schneider [4] characterize the "good thing" in a liveness property as

- *always possible*, no matter what has occurred so far, and
- *possibly infinite*, so it need not be a discrete event.

For example, guaranteed service GS (2.3) is a liveness property in which the good thing is the eventual response to a request. This good thing is always possible, because a state in which a response is produced can always be appended to any finite trace containing a request. And this good thing is not infinite because the response is a discrete event, but *starvation freedom*, which stipulates that a system makes progress infinitely often, is an example of a liveness property with an infinite good thing.

Formally, a good thing is an infinite suffix of a finite trace:

Liveness property. Trace property L is a *liveness property* [4] iff

$$(\forall\, t \in \Psi_{\mathsf{fin}} : (\exists\, t' \in \Psi_{\mathsf{inf}} : t \leq t' \;\wedge\; t' \in L)).$$

Define LP to be the set of all liveness properties. Not surprisingly, LP is a hyperproperty.

Just as with hypersafety, we generalize liveness to hyperliveness by generalizing a finite trace to a finite set of finite traces. The definition of hyperliveness is essentially the same as the definition of liveness, except for an additional level of sets:

Liveness hyperproperty. Hyperproperty ***L*** is a *liveness hyperproperty* (is *hyperliveness*) iff

$$(\forall\, T \in \mathsf{Obs} : (\exists\, T' \in \mathsf{Prop} : T \leq T' \;\wedge\; T' \in \boldsymbol{L})).$$

Define $\mathbf{\mathsf{LHP}}$ to be the set of all liveness hyperproperties.

Mean response time ***RT*** (2.7) is not liveness but it is hyperliveness: the good thing is that the mean response time is low enough. Given any observation T with any mean response time, it is always possible to extend T, such that the resulting system has a low enough mean response time, by adding a trace that has many quick responses. Note that if this policy were approximated by limiting the maximum response time in each execution, then the resulting hyperproperty would be a lifted safety property.

Some additional consequences of the definition of hyperliveness are:

- The only hyperproperty that is both hypersafety and hyperliveness is ***true***, defined as Prop. The hyperproperty ***false***, defined as $\{\emptyset\}$, is hypersafety but not hyperliveness.[16]

- Liveness properties lift to liveness hyperproperties.

  Proposition 2. $(\forall\, L \in \mathsf{Prop} : L \in \mathsf{LP} \iff [L] \in \mathsf{LHP})$.

- Set LP of all liveness properties is a liveness hyperproperty: every observation can be extended to any liveness property.

- Similarly, set SP of all safety properties is a liveness hyperproperty: every observation can be extended to a safety property (whose bad thing is "not beginning execution with one of the finite traces in the observation").

Possibilistic information flow. Some information-flow security policies, such as observational determinism ***OD*** (2.6), restrict nondeterminism of a system from being publicly observable. However, observable nondeterminism might be useful, for a couple of reasons. First, systems might exhibit nondeterminism due to scheduling. If the scheduler cannot be influenced by secret information (i.e., the scheduler does not serve as a covert timing channel), then it is reasonable to allow the scheduler to behave nondeterministically. Second, nondeterminism is a useful modeling abstraction when dealing with probabilistic systems (which we consider in more detail in section 7.2.4). When the exact probabilities for a system are unknown, they can be abstracted by nondeterminism. For at least these reasons, there is a history of research on *possibilistic* information-flow security policies, beginning with nondeducibility [59] and generalized noninterference [40]. Such policies are founded on the intuition that low observers of a system should gain little from their observations. Typically, these policies require that every low observation is consistent with some large set of possible high behaviors.

McLean [42] shows that possibilistic information-flow policies can be expressed as trace sets that are closed with respect to *selective interleaving functions*. Such functions, given two executions of a system, specify another trace that must also be an execution of the system—as did the definition of generalized noninterference ***GNI*** (2.5). Mantel [38] generalizes from these functions to *closure operators*, which extend a set S of executions to a set S' such that $S \subseteq S'$. Mantel argues that every possibilistic information-flow policy can be expressed as a closure operator.

Given a closure operator Cl that expresses a possibilistic information-flow policy, the hyperproperty $\boldsymbol{P}_{Cl}$ induced by Cl is

$$\boldsymbol{P}_{Cl} \;\triangleq\; \{Cl(T) \mid T \in \mathsf{Prop}\}.$$

Define the set PIF of all such hyperproperties to be $\bigcup_{Cl} \boldsymbol{P}_{Cl}$. It is now easy to see that these are liveness hyperproperties: any observation T can be extended to its closure.

Theorem 3. $\mathsf{PIF} \subset \mathsf{LHP}$.

[16]The false property is the empty set of traces, so it might seem reasonable to define ***false*** as the empty set of trace sets. But then the lift of the false property would not equal ***false***. Note that ***false*** is not satisfied by any system because, by definition, $\emptyset$ is not a system.

Possibilistic information-flow policies are therefore never hypersafety.[17]

Temporal logics. Consider the hyperproperty "For every initial state, there is some terminating trace, but not all traces must terminate," denoted as ***NNT***. In branching-time temporal logic, ***NNT*** could be expressed as

$$\lozenge\ terminates, \tag{5.1}$$

where $terminates$ is a state predicate and $\lozenge$ is the "not never" operator.[18] There is no linear-time temporal predicate that expresses ***NNT***, nor is there a liveness property equivalent to ***NNT*** [34]; an approximation would be a linear-time predicate, or a liveness property, that requires every trace to terminate. However, ***NNT*** is hyperliveness because any finite trace can be extended to a set of executions such that at least one execution terminates.

This example suggests a relationship between hyperproperties and branching-time temporal predicates, and between trace properties and linear-time temporal predicates. We can make this relationship precise by examining the semantics of temporal logic. In both branching time and linear time, a semantic model contains a set of states and a valuation function assigning a Boolean value to each atomic proposition in each state. Additionally, a branching-time model requires a current state and a set of traces, whereas a linear-time model requires a single trace [21]. These requirements differ because a linear-time predicate is a property of a trace, whereas a branching-time predicate is a property of a state and all the future traces that could proceed from that state. Thus, trace properties model linear-time predicates, and hyperproperties model branching-time predicates for a given state.

Moreover, hyperproperties can express policies that branching-time predicates cannot. Consider the trace property "Every trace must end with an infinite number of $good$ states," denoted SAG, where $good$ is a state predicate. In linear-time temporal logic, SAG could be expressed as

$$\leadsto \Box\ good, \tag{5.2}$$

where $\leadsto$ is the "sometime" operator and $\Box$ is the "always" operator. SAG is liveness and thus hyperliveness, but there is no branching-time predicate that expresses it [34].

6 Topology

Topology enables an elegant characterization of the structure of hyperproperties, just as it did for trace properties. We begin by summarizing the topology of trace properties [58].

Consider an *observer* of an execution of a system, who is permitted to see each new state as it is produced by the system; otherwise, the system is a black box to the observer. The observer attempts to determine whether trace property P holds of the system. At any point in time, the observer has seen only a finite prefix of the (infinite)

[17]Another way to reach this conclusion is to observe that closure operators need not yield hyperproperties that are subset closed—yet, by Theorem 1, every safety hyperproperty is subset closed.

[18]Temporal logic CTL [13] would express this formula as $\mathsf{E}\ \mathsf{F}\ terminates$.

execution. Thus, the observer should declare that the system satisfies P, after observing finite trace t, only if all possible extensions of t will also satisfy P. Abramsky names such properties *observable* [3].

Like the bad thing for a safety property, a observable property must be detectable in finite time; and once detected, hold thereafter. Formally, O is a observable property iff

$$(\forall\, t \in \Psi_{\mathsf{inf}} : t \in O \implies (\exists\, m \in \Psi_{\mathsf{fin}} : m \leq t \\ \wedge\ (\forall\, t' \in \Psi_{\mathsf{inf}} : m \leq t' \implies t' \in O))).$$

Define $\mathcal{O}$ to be the set of observable properties. This set satisfies two closure conditions. First, if $O_1, \ldots, O_n$ are observable, then $\bigcap_{i=1}^{n} O_i$ is also observable. Second, if $\boldsymbol{O}$ is a (potentially infinite) set of observable properties, then $\bigcup_{O \in \boldsymbol{O}} O$ is also observable. Thus $\mathcal{O}$ is *closed under finite intersections and infinite unions*.

A *topology* on a set S is a set $\mathcal{T} \subseteq \mathcal{P}(S)$ such that $\mathcal{T}$ is closed under finite intersections and infinite unions. Because $\mathcal{O}$ is so closed, it is a topology on Ψ_{inf}. We name $\mathcal{O}$ the *Plotkin* topology, because Plotkin proposed its use in characterizing safety and liveness [4].[19]

The elements of a topology $\mathcal{T}$ are called its *open sets*. A convenient way to characterize the open sets of a topology is in terms of a base or a subbase. A *base* of topology $\mathcal{T}$ is a set $\mathcal{B} \subseteq \mathcal{T}$ such that every open set is a (potentially infinite) union of elements of $\mathcal{B}$. A *subbase* is a set $\mathcal{A} \subseteq \mathcal{T}$ such that the collection of finite intersections of $\mathcal{A}$ is a base for $\mathcal{T}$. The set

$$\mathcal{O}^B \ \triangleq\ \{\uparrow t \mid t \in \Psi_{\mathsf{fin}}\}$$

is a base (and a subbase) of the Plotkin topology, where

$$\uparrow t \ \triangleq\ \{t' \in \Psi_{\mathsf{inf}} \mid t \leq t'\}$$

is the *completion* of a finite trace t. When $t \leq t'$ we say that t' *extends* t. The completion of t is thus the set of all infinite extensions of t.

Alpern and Schneider [4] noted that, in the Plotkin topology, safety properties correspond to closed sets and liveness properties correspond to dense sets. A *closed* set is the complement (with respect to S) of an open set. If a trace t is not a member of a closed set C, then there is some bad thing (specifically, the prefix m of t in the definition of observable as instantiated on open set $\overline{C}$, the complement of C) that is to blame; the existence of such bad things makes C a safety property. Likewise, a set that is *dense* intersects every non-empty open set in $\mathcal{T}$. So for any finite trace t and dense set D, the intersection of $\uparrow t$ (which is open because it is a member of $\mathcal{O}^B$) and D is nonempty. Since any finite trace can be extended to be in D, it holds that D is a liveness property.

We want to construct a topology on sets of traces that extends this correspondence to hyperproperties. The most important step is generalizing the notion of finite observability from trace properties to hyperproperties. Section 3 already did this in generalizing a finite trace to a finite set of finite traces—that is, an observation. The observer,

[19] Topology $\mathcal{O}$ is also the Scott topology on the ω-algebraic CPO of traces ordered by $\leq$ [58].

as before, sees the system produce each new state in the execution. However, the observer may now reset the system at any time, causing it to begin a new execution. At any finite point in time, the observer has now collected a finite set of finite (thus partial) executions. An observation is thus an element of Obs, as defined in section 3.

An extension of an observation should allow the observer to perform additional resets of the system, yielding a larger set of traces. An extension should also allow each execution to proceed longer, yielding longer traces. So extension corresponds to trace set prefix $\leq$ (cf. section 3). The completion of observation M is

$$\uparrow M \;\triangleq\; \{T \in \mathsf{Prop} \mid M \leq T\}.$$

We can now define our topology on sets of traces in terms of its subbase:

$$\mathcal{O}^{SB} \;\triangleq\; \{\uparrow M \mid M \in \mathsf{Obs}\}.$$

The base $\mathcal{O}^B$ of our topology is then $\mathcal{O}^{SB}$ closed under finite intersections. The base and subbase turn out to be the same sets.

Proposition 3. $\mathcal{O}^B = \mathcal{O}^{SB}$.

Finally, our topology $\mathcal{O}$ is $\mathcal{O}^B$ closed under infinite unions.

Define $\mathcal{C}$ to be the closed sets in our topology and $\mathcal{D}$ to be the dense sets. Just as safety and liveness correspond to closed and dense sets in the Plotkin topology, hypersafety and hyperliveness correspond to closed and dense sets in our generalization of that topology.

Proposition 4. $\mathsf{SHP} = \mathcal{C}$.

Proposition 5. $\mathsf{LHP} = \mathcal{D}$.

Our topology $\mathcal{O}$ is actually equivalent to well-known topology, as stated by the following theorem. The *Vietoris* (or *finite* or *convex Vietoris*) topology is a standard construction of a topology on sets out of an underlying topology [43, 61]. Our underlying topology was on traces, and we constructed a topology on sets of traces. The Vietoris construction can be decomposed into the *lower Vietoris* and *upper Vietoris* constructions [57], which also yield topologies. Let $\mathfrak{V}_L(\mathcal{T})$ denote the lower Vietoris construction, which given underlying topology $\mathcal{T}$ on space $\mathcal{X}$ produces the topology on $\mathcal{P}(\mathcal{X})$ induced by subbase $\mathfrak{V}_L^{SB}(\mathcal{T})$:

$$\mathfrak{V}_L^{SB}(\mathcal{T}) \;\triangleq\; \{\langle O \rangle \mid O \in \mathcal{T}\},$$

where $\langle T \rangle$ is defined[20] as follows:

$$\langle T \rangle \;\triangleq\; \{U \in \mathcal{P}(\mathcal{X}) \mid U \cap T \neq \emptyset\}.$$

The following theorem states that our topology is equivalent to the lower Vietoris construction applied to the Plotkin topology:

[20]Operators $[\cdot]$ (from section 2) and $\langle \cdot \rangle$ are similar to modal logic operators $\Box$ (necessity) and $\Diamond$ (possibility): For trace property T, lift $[T]$ denotes the set of all refinements of T—that is, the hyperproperty in which T is necessary. Similarly, $\langle T \rangle$ denotes the set of all trace properties that share a trace with T—that is, the hyperproperty in which T is always possible.

Theorem 4. $\boldsymbol{\mathcal{O}} = \mathfrak{V}_L(\mathcal{O})$.

Smyth [57] established that the lower Vietoris topology is equivalent to the *lower* (or *Hoare*) *powerdomain*, which is a construction used to model the semantics of nondeterminism [48]. So our topology embodies the same intuition about nondeterminism as the lower powerdomain does.

The proof of Theorem 4 yields another topological characterization of safety hyperproperties: the set of lifted safety properties, closed under infinite intersections and finite unions (denoted as closure operator Cl_C, because these closure conditions characterize a *topology of Closed sets*), is the set of safety hyperproperties.

Proposition 6. $\mathsf{SHP} = Cl_C(\{[S] \mid S \in \mathsf{SP}\})$.

Defining trace set prefix. Recall that trace set prefix $\leq$ is defined as follows:

$$T \leq T' \;\triangleq\; (\forall\, t \in T : (\exists\, t' \in T' : t \leq t')).$$

For clarity, we use $\leq_L$ instead of $\leq$ to refer to that definition throughout the rest of this section (L stands for Lower Vietoris).

Two natural alternatives to $\leq_L$ are

$$\begin{aligned} T \leq_U T' \;&\triangleq\; (\forall\, t' \in T' : (\exists\, t \in T : t \leq t')), \\ T \leq_C T' \;&\triangleq\; T \leq_L T' \wedge T \leq_U T'. \end{aligned}$$

(U and C stand for Upper and Convex Vietoris. These prefix relations correspond to the eponymous topologies.) However, both alternatives turn out to be unsuitable for our purposes, because they do not correspond to our intuition about finite observability—as we now explain.

Hyperproperty $\boldsymbol{O}$ is observable iff

$$\begin{aligned} (\forall T \in \mathsf{Prop} : T \in \boldsymbol{O} \implies (\exists\, M \in \mathsf{Obs} : M \leq T \\ \wedge\ (\forall T' \in \mathsf{Prop} : M \leq T' \implies T' \in \boldsymbol{O}))). \end{aligned}$$

Consider using $\leq_U$ for trace set prefix $\leq$. For a concrete example, suppose that $\Sigma = \{a, b, c\}$, $\boldsymbol{O}$ is observable, $T \in \boldsymbol{O}$, and $M = \{a, b\}$. Any T' such that $M \leq_U T'$ must be a member of $\boldsymbol{O}$. Every trace t' in T' must begin with either a or b and cannot begin with c. In particular, T' might contain traces beginning only with b, never with a. Observation M therefore characterizes a system in which a nondeterministic choice to produce c as the first state is not possible. So with $\leq_U$, an observation records what nondeterminism is denied, and all future extensions of that observation are also required to deny that nondeterminism.

In contrast, with $\leq_L$ (i.e., our topology), an observation records what nondeterminism has so far been permitted, and all future extensions of that observation are required also to permit that nondeterminism. Our intuition is that observers of a black-box system can observe permitted nondeterminism (by observing states produced by the system) but not denied nondeterminism. The definition of $\leq_U$ does not correspond to that intuition, but the definition of $\leq_L$ does. Similarly, using $\leq_C$ for trace set prefix leads to observations that record both permitted and denied nondeterminism (because $\leq_C$ is the conjunction of $\leq_L$ and $\leq_U$), and therefore $\leq_C$ does not correspond to our intuition, either.

So neither the upper nor the convex Vietoris topology enjoys open sets that are the observable hyperproperties; consequently, the equivalence of closed sets and hypersafety is lost. Nonetheless, these topologies might be useful for other purposes—for example, in refusal semantics for CSP [30].

7 Beyond Hypersafety and Hyperliveness

7.1 Intersections

Security policies can exhibit features of both safety and liveness. For example, consider a policy on a medical information system that must maintain the confidentiality of patient records and must also eventually notify patients whenever their records are accessed [6]. If the confidentiality requirement is interpreted as observational determinism ***OD*** (2.6), then this system must both prevent bad things (***OD***, which is hypersafety) as well as guarantee good things (eventual notification, which can be formulated as liveness). As another example, consider an asynchronous proactive secret-sharing system [67] that must maintain and periodically refresh a secret. Each share refresh must complete during a given time interval with high probability. Maintaining the confidentiality of the secret can be formulated as ***SecS*** (4.1), which is hypersafety. The eventual refresh of the secret shares can be formulated as liveness: every execution eventually completes the refresh if enough servers remain uncompromised. And the high probability that the refresh succeeds within a given time interval is hyperliveness—similar to mean response time ***RT*** (2.7). Both of these examples illustrate hyperproperties that are intersections of (hyper)safety and (hyper)liveness.

In fact, as stated by the following theorem, every hyperproperty is the intersection of a safety hyperproperty with a liveness hyperproperty. This theorem generalizes the result of Alpern and Schneider [4] that every trace property is the intersection of a safety property and a liveness property.

Theorem 5. $(\forall \boldsymbol{P} \in \mathsf{HP} : (\exists \boldsymbol{S} \in \mathsf{SHP}, \boldsymbol{L} \in \mathsf{LHP} : \boldsymbol{P} = \boldsymbol{S} \cap \boldsymbol{L}))$.

7.2 System Representations

Recall that hyperproperties are system properties in which system execution is modeled with trace sets. Some models of system execution are expressed with other mathematical formalisms—for example, Goguen and Meseguer's noninterference ***GMNI*** (2.4) models systems as deterministic state machines.

We have not yet classified ***GMNI*** as hypersafety or hyperliveness. Recall that our formalization of ***GMNI*** included the conjunct "$T \in$ ***SM***", where hyperproperty ***SM*** is the set of all trace sets that encode deterministic state machines. Therefore ***GMNI*** excludes all trace sets that do not encode a deterministic state machine. It is reasonable to expect that ***GMNI*** is hypersafety; the bad thing is a set $\{t, t'\}$ of finite traces where t' contains no high inputs and contains the same low inputs as t, yet t and t' have different low outputs. But ***GMNI*** fails to be hypersafety because of a technicality—a system T might fail to satisfy ***GMNI*** only because T is nondeterministic, in which case a deterministic, non-interfering observation of T would be remediable hence ***GMNI*** would not be hypersafety.[21] The problem is that the definition of hypersafety, by quantifying over Prop, assumed that systems are allowed to be nondeterministic. Now that we are interested in a restricted system representation, we need to restrict the definition of hypersafety and quantify over a smaller set of systems. Let Rep be a set of trace sets denoting a system representation—that is, a subset of Prop containing those trace sets that represent systems of interest. And let $\mathit{Obs}(\mathsf{Rep})$ denote the subset of Obs containing observations of Rep, where $\mathit{Obs}(\mathsf{Rep}) = \{M \in \mathsf{Obs} \mid (\exists T \in \mathsf{Rep} : M \leq T)\}$. Now we can define hypersafety and hyperliveness for a given system representation.

Safety hyperproperty for system representation Rep. A hyperproperty $\boldsymbol{S}$ is a *safety hyperproperty for system representation* Rep (is *hypersafety for* Rep) iff

$$\begin{aligned}(\forall T \in \mathsf{Rep} : T \notin \boldsymbol{S} \implies (\exists M \in \mathit{Obs}(\mathsf{Rep}) : M \leq T \\ \wedge\ (\forall T' \in \mathsf{Rep} : M \leq T' \implies T' \notin \boldsymbol{S}))).\end{aligned}$$

Liveness hyperproperty for system representation Rep. Hyperproperty $\boldsymbol{L}$ is a *liveness hyperproperty for system representation* Rep (is *hyperliveness for* Rep) iff

$$(\forall T \in \mathit{Obs}(\mathsf{Rep}) : (\exists T' \in \mathsf{Rep} : T \leq T' \ \wedge\ T' \in \boldsymbol{L})).$$

GMNI indeed is hypersafety for ***SM***, fulfilling our expectation.

The results proved in this paper about hypersafety and hyperliveness generalize naturally to system representation besides Prop. Informally, the generalizations are as follows:[22]

- If P is safety (liveness) for Rep, then $[P]$ is hypersafety (hyperliveness) for Rep (generalizing Propositions 1 and 2).
- If $\boldsymbol{P}$ is hypersafety for Rep, then $\boldsymbol{P}$ is subset closed for Rep, but not necessarily subset closed for Prop (generalizing Theorem 1). Consequently, stepwise refinement does not necessarily work with hyperproperties that are hypersafety for Rep.
- If $\boldsymbol{P}$ is a possibilistic information-flow policy for Rep, then $\boldsymbol{P}$ is hyperliveness for Rep (generalizing Theorem 3).
- k-hypersafety for Rep can be reduced to safety for Rep^k (generalizing Theorem 2).
- Every hyperproperty for Rep is the intersection of a safety hyperproperty for Rep with a liveness hyperproperty for Rep (generalizing Theorem 5).

[21] A similar problem would occur even if we used implication instead of conjunction to formalize the requirement that systems be deterministic state machines: any observation could be remediated by adding traces that represent nondeterministic transitions of the state machine.

[22] We leave generalizing the topological results as future work. However, since the intersection theorem generalizes, we believe that the topological results could also be generalized.

Appendix C gives formal statements of these results. The proofs of these results are all straightforward corollaries of the original results, although some proofs require additional assumptions about Rep.

We now explore system properties in various system representations—relational systems, labeled transition systems, state machines, and probabilistic systems—by encoding each into trace sets, thus into hyperproperties.

7.2.1 Relational Systems

In language-based information-flow security [53], a program P is sometimes modeled (e.g., with large-step operational semantics) as a relation $\Downarrow$ such that $\langle P, s\rangle \Downarrow s'$ if P begun in *initial* state s terminates in *final* state s'. Using this relation, noninterference can be stated as

$$s_1 =_L s_2 \wedge \langle P, s_1\rangle \Downarrow s_1' \wedge \langle P, s_2\rangle \Downarrow s_2' \implies s_1' =_L s_2',$$

where relation $=_L$ (cf. observational determinism ***OD*** (2.6)) determines which states are low-equivalent. This statement of noninterference is *termination insensitive* because it allows information to leak through termination channels.

To model a program P as set T of traces, intuitively, imagine that an observer of the program periodically checks to see in what state the program is. If P begun in initial state s never terminates, then the observer will see an infinite sequence containing only s. If P does terminate in final state s', then the observer will see a finite sequence of s followed by an infinite sequence of s'. Let T be the set of all such traces. Formally, T is defined as follows:

$$\begin{aligned} T \;=\; &\{t \in \Psi_{\mathsf{inf}} \mid \langle P, s\rangle \Downarrow s' \wedge t \in s^+(s')^\omega\} \\ &\qquad \cup \{t \in \Psi_{\mathsf{inf}} \mid \neg(\exists\, s' : \langle P, s\rangle \Downarrow s') \wedge t = s^\omega\}. \end{aligned}$$

Let ***Rel***, the set of all *relational systems*, be the set of all trace sets so constructed for any P.

Define *termination-insensitive relational noninterference* as a hyperproperty:

$$\begin{aligned} \textbf{\textit{TIRNI}} \;\triangleq\; &\{T \in \mathsf{Prop} \mid T \in \textbf{\textit{Rel}} \\ &\quad \wedge (\forall\, t_1, t_2 \in T : t_1[0] =_L t_2[0] \\ &\qquad \implies \mathit{diverges}(t_1) \vee \mathit{diverges}(t_2) \\ &\qquad\qquad \vee (\exists\, s_1, s_2 \in \Sigma : \mathit{terminates}(t_1, s_1) \\ &\qquad\qquad\quad \wedge \mathit{terminates}(t_2, s_2) \wedge s_1 =_L s_2))\}. \end{aligned} \tag{7.1}$$

Predicate $\mathit{diverges}(t)$ holds whenever t is a trace of a program P such that P does not terminate when begun in initial state $t[0]$, so $t = (t[0])^\omega$. Similarly, predicate $\mathit{terminates}(t, s)$ holds whenever P terminates in final state s when begun in initial state $t[0]$, so $t = (t[0])^+ s^\omega$. We assume without loss of generality that final states are distinguishable from initial states (e.g., by having a special flag set), so that $\mathit{diverges}$ and $\mathit{terminates}$ can distinguish between nontermination and termination in a final state that otherwise is identical to an initial state. ***TIRNI*** is hypersafety for ***Rel***: the bad thing is a pair of traces that begin in low-equivalent initial states but terminate in final states that are not low-equivalent.

Termination-sensitive noninterference is the same as termination insensitive, except that it forbids one trace to diverge and the other to terminate. So define *termination-sensitive relational noninterference* as follows:

$$\begin{aligned}\textbf{\textit{TSRNI}} \triangleq \{T \in \mathsf{Prop} \mid\ & T \in \textbf{\textit{Rel}} \\ & \wedge (\forall\, t_1, t_2 \in T : t_1[0] =_L t_2[0] \\ & \quad \Longrightarrow (\mathit{diverges}(t_1) \wedge \mathit{diverges}(t_2)) \\ & \qquad \vee (\exists\, s_1, s_2 \in \Sigma : \mathit{terminates}(t_1, s_1) \\ & \qquad\quad \wedge \mathit{terminates}(t_2, s_2) \wedge s_1 =_L s_2))\}. \end{aligned} \tag{7.2}$$

Note that the only change is that a disjunction became a conjunction. ***TSRNI*** is not hypersafety for ***Rel***: A system containing a pair $\{t, t'\}$ of traces, where t diverges and t' does not, yet where t and t' contain low-equivalent initial states, does not satisfy ***TSRNI***. But any finite prefix of this pair could be remediated by extending the prefix of t to terminate in the same final state as t'. Likewise, ***TSRNI*** is not hyperliveness for ***Rel***:[23] Consider a finite observation containing a pair of terminating traces that have low-equivalent initial states but not low-equivalent final states. This observation cannot be extended to be in ***TSRNI***.

7.2.2 Labeled Transition Systems

Definitions of noninterference are sometimes based on *bisimulation*, which is a relation that specifies whether two systems are equivalent to an observer. Bisimulations are often expressed over *labeled transition systems*, which are triples $(S, L, \rightarrow)$ where S is a set of LTS-states,[24] L is a set of labels, and $\rightarrow$ is a relation on $S \times L \times S$ [45]. Elements of relation $\rightarrow$ are usually notated $s_1 \xrightarrow{\ell} s_2$ and are interpreted to mean that the system has a transition labeled ℓ from LTS-state s_1 to LTS-state s_2.

A labeled transition system $(S, L, \rightarrow)$ can be encoded as a set of traces. Define the state space Σ for the traces to be $S \times L$.[25] Given state $s \in \Sigma$, let $st(s)$ denote the LTS-state from s, and let $lab(s)$ denote the label from s. Define $traces(S, L, \rightarrow)$ to be

$$\{t \mid (\forall\, i \in \mathbb{N} : st(t[i]) \xrightarrow{lab(t[i])} st(t[i+1]))\}.^{26}$$

Let ***LTS*** be the set of all trace sets so constructed for any LTS.

[23]Terauchi and Aiken [60] characterized termination-sensitive noninterference as "2-liveness," where they defined "2-liveness" as a "property which may observe up to two possibly infinite traces to refute the property." Although they are correct that ***TSRNI*** could be refuted by observing two infinite traces, refutation is really about safety, not liveness—there is no good thing for ***TSRNI***, but there is an infinitely-observable bad thing. So "2-infinite-safety" would be a better term than "2-liveness."

[24]We use the term *LTS-state* to distinguish these from the states defined in section 2.

[25]This construction would not work with an impoverished notion of state, as observed by Focardi and Gorrieri [22] for states that are elements only of L.

[26]We could replace $lab(t[i])$ with $lab(t[i+1])$ in this definition; the choice of where to store the label is arbitrary.

We now demonstrate how to use this encoding by formalizing Focardi and Gorrieri's [22] definition of *bisimulation nondeducibility on compositions* (BNDC), which is a noninterference policy for nondeterministic LTSs. The intuition behind this policy is that a system should appear the same to a low observer no matter with what other system it is composed (i.e., run in parallel). Assume that set L of labels can be partitioned into three sets of *actions* (i.e., events): a set of low security actions, a set H of high security actions, and $\{\tau\}$, where τ is an unobservable *internal* action. An LTS $E = (S, L, \rightarrow)$ satisfies BNDC, denoted $\mathit{BNDC}(E)$, iff for all LTSs $F = (S, H \cup \{\tau\}, \rightarrow_F)$ that take only high and internal actions,

$$E/H \approx (E|F) \setminus H,$$

with notations $/$, $|$, $\setminus$, and $\approx$ informally defined as follows:[27]

- Hiding operator E/H relabels as τ all actions from H that occur during execution of E. System E/H thus represents the view of system E by a low observer, since all the high actions are hidden.
- Parallel composition operator $E|F$ denotes the interleaving of systems E and F. The systems can synchronize on actions, causing the composed system to emit internal action τ.
- Restriction operator $E \setminus H$ prohibits the occurrence of any actions from H during execution of E, meaning that no transition with a label from H is allowed. System $(E|F) \setminus H$ thus represents a low observer's view of E when all the high actions that E takes are synchronized with F.
- Weak bisimulation relation $E \approx F$ intuitively means that E and F can simulate each other: if E can take a transition with label ℓ, then there must exist a transition of F that is also labeled ℓ, and after taking those transitions E and F must remain bisimilar. F is allowed to take any number of internal transitions (labeled τ) before or after the ℓ-labeled transition. Further, the relation must be symmetric, such that if $E \approx F$ then $F \approx E$.

Thus, if $E/H \approx (E|F) \setminus H$, then a low observer's view of E does not change when E is composed with any high security system F. The hyperproperty corresponding to Focardi and Gorrieri's BNDC is

$$\begin{aligned}\boldsymbol{\mathit{BNDC}} \triangleq \{T \in \mathsf{Prop} \mid T \in \boldsymbol{\mathit{LTS}} & \\ \wedge\ (\exists\, E \in \boldsymbol{\mathit{LTS}} : T = \mathit{traces}(E) & \\ \wedge\ \mathit{BNDC}(E))\}. & \quad (7.3)\end{aligned}$$

BNDC is hyperliveness for ***LTS*** because of the existential in definition of $\approx$: any observation can be remedied by adding additional transitions. This remediation corresponds to a closure operator because it only adds traces, thus ***BNDC*** is a possibilistic-information flow policy.

Appendix B presents another bisimulation-based noninterference policy as a hyperproperty.

[27]The formal definitions (over LTSs) are standard and given by Focardi and Gorrieri [22]. It is straightforward to define them directly over trace sets.

7.2.3 State Machines

Goguen and Meseguer [23] define a *state machine* as a tuple (S, C, O, out, do, s_0), where S is a set of machine states, C is a set of commands, O is a set of outputs, out is a function from S to O yielding what output the user of the machine observes when the machine is in a given state, do is a function from $S \times C$ to S describing how the machine transitions between states as a function of commands, and s_0 is the initial state of the machine.[28] Such state machines are deterministic because do is a function rather than a relation.

A state machine $M = (S, C, O, out, do, s_0)$ can be encoded as a set of traces. The construction proceeds in two steps. First, M is encoded as a labeled transition system (cf. section 7.2.2) by treating the machine commands and outputs as labels: Let the set $\hat{S}$ of LTS-states be set S of machine states. Let the set $\hat{L}$ of labels be product set $C \times O$ of commands and outputs. Let the transition relation $\rightarrow$ include $(s, (c, o), s')$ whenever $do(s, c) = s'$ and $out(s') = o$. We now have a labeled transition system $L = (\hat{S}, \hat{L}, \rightarrow)$. Second, the traces of M are the traces of L that start with s_0: let $traces(M)$ be $traces(\hat{S}, L, \rightarrow) \cap \{t \in \Psi_{\mathsf{inf}} \mid t[0] = s_0\}$.

The set ***SM*** of all state machines is a hyperproperty:

$$\textbf{\textit{SM}} \triangleq \{T \in \mathsf{Prop} \mid (\exists\, M : T = traces(M))\}. \tag{7.4}$$

As noted at the beginning of this section, ***GMNI*** is hypersafety for ***SM***.

7.2.4 Probabilistic Systems

A *probabilistic system* is equipped with a function p such that the system transitions from a state s to state s' with probability $p(s, s')$.[29] This probability is *Markovian* because it does not depend upon past or future states in an execution; nonetheless, dependence upon the past or future can be modeled by allowing states to contain history or prophecy variables [1]. Function p can itself even be encoded into the state in various ways. For example, state s could record $p(s, s')$ for all states s'. Or in a trace t, state $t[i]$ could record $p(t[i], t[i+1])$. This latter encoding is an instantiation of the construction in section 7.2.2 for encoding labeled transition systems as sets of traces; here, the labels are probabilities. Either way, probabilistic systems can be modeled as sets of traces. Define ***PR*** to be the set of all trace sets that encode probabilistic systems—that is, trace set T is in ***PR*** if T encodes a valid probability function $p(\cdot, \cdot)$.

To obtain a probability measure on sets of traces, let $\mathsf{Pr}_{s,S}(T)$ denote the probability with which set T of finite traces is produced by probabilistic system S beginning in initial state s.[30] O'Neill et al. [47] show how to construct this probability measure from p. We now demonstrate how the measure can be used in the definitions of hyperproperties.

[28]Our definition of state machines simplifies Goguen and Meseguer's by omitting user clearances, though the clearances still appear in the definition of ***GMNI***.

[29]To be a valid probability, $p(s, s')$ must be in the real interval [0,1] for all s and s'; and for all s, it must hold that $\sum_{s'} p(s, s') = 1$.

[30]The initial state can be eliminated if we also assume a prior probability on initial states [26, §6.5]. The requirement that the traces in T be finite is, however, essential to ensure that $\mathsf{Pr}_{s,S}(T)$ is a valid probability measure.

Probabilistic noninterference. In information-flow security, the original motivation for adding probability to system models was to address covert channels and to establish connections between information theory and information flow [24,25,44]. *Probabilistic noninterference* [25] emerged from this line of research. Intuitively, this policy requires that the probability of every low trace be the same for every low-equivalent initial state. To formulate probabilistic noninterference as a hyperproperty, we need some notation. Let the *low equivalence class* of a finite trace t be denoted $[t]_L$, where

$$[t]_L \triangleq \{t' \in \Psi_{\mathsf{fin}} \mid ev_L(t) = ev_L(t')\}.$$

The probability that system S, starting in state s, produces a trace that is low-equivalent to t is therefore $\mathsf{Pr}_{s,S}([t]_L)$. Let the set of initial states of trace property T be denoted $\mathit{Init}(T)$, where

$$\mathit{Init}(T) \triangleq \{s \mid \{s\} \leq T\}.$$

Probabilistic noninterference can now be expressed as follows:

$$\begin{aligned} \boldsymbol{\mathit{PNI}} \triangleq \{T \in \mathsf{Prop} \mid {} & T \in \boldsymbol{\mathit{PR}} \\ & \wedge (\forall s_1, s_2 \in \mathit{Init}(T) : ev_L(s_1) = ev_L(s_2) \\ & \qquad \Longrightarrow (\forall t \in \Psi_{\mathsf{fin}} : \mathsf{Pr}_{s_1,T}([t]_L) = \mathsf{Pr}_{s_2,T}([t]_L)))\}. \end{aligned} \quad (7.5)$$

PNI is not hyperliveness for ***PR***, because a system that deterministically produces two non-low-equivalent traces from two initial low-equivalent states cannot be extended to satisfy ***PNI***. Whether ***PNI*** is hypersafety for ***PR*** depends on whether state space Σ is finite. To see why, consider a system T such that $T \notin$ ***PNI*** and $T \in$ ***PR***. We can attempt to construct a bad thing M for T as follows. Since $T \notin$ ***PNI***, there exists a trace t_L of low events that is produced by initial states s_1 and s_2 with differing probabilities. Let M be the prefix of T that completely determines the probability of t_L for those initial states:

$$M = \{t \in \Psi_{\mathsf{fin}} \mid t[0] \in \{s_1, s_2\} \wedge t \leq T \wedge ev_L(t) = t_L\}.$$

Recall that bad things must be finitely observable and irremediable. M is irremediable because no extension of it can change the probability of t_L for initial states s_1 and s_2. But is M finitely observable—that is, is $M \in \mathsf{Obs}$? Recall that an element of Obs must be a finite set of finite traces. Each trace in M is finite, but M might not be a finite set:

- If state space Σ is countably infinite,[31] then there could be infinitely many states to which s_1 (and s_2) transition. Hence there could need to be infinitely many traces in M to completely determine the probability of t_L, so M could not be in Obs. Moreover, any finite subset N of M would necessarily omit some states from Σ. So it might be possible to extend N to a system T' that satisfies ***PNI*** by adding traces containing those omitted states. Thus T would have no bad thing, and ***PNI*** would not be hypersafety for ***PR***.

[31] State space Σ cannot be uncountably infinite without generalizing probability function $p(\cdot,\cdot)$ to a probability measure.

- If Σ is finite, then only finitely many finite traces are low-equivalent to t_L. Thus M is finite, and no extension of T' of M can change the probability of t_L. So T' cannot be in ***PNI***. Therefore ***PNI*** is hypersafety for ***PR***.

Gray's definition of probabilistic noninterference [25] is hypersafety for ***PR***, because Gray required the state (and input and output) space to be finite. But the definition of O'Neill et al. [47] is neither hypersafety nor hyperliveness, because it allowed a countably infinite state space.

Secure encryption. A *private-key encryption scheme* is a tuple ($\mathcal{M}$, $\mathcal{K}$, $\mathcal{C}$, *Gen*, *Enc*, *Dec*), where $\mathcal{M}$ is the *message space*, $\mathcal{K}$ is the *key space*, and $\mathcal{C}$ is the *ciphertext space*, such that the following hold:

- Gen is the *key-generation algorithm*, a randomized algorithm that produces a key $k \in \mathcal{K}$. We write $k \leftarrow \mathit{Gen}$ to denote the sampling of k from the probability distribution induced by Gen.

- Enc is the *encryption algorithm*, an algorithm (either randomized or deterministic) that accepts a key $k \in \mathcal{K}$, a plaintext message $m \in \mathcal{M}$, and yields a ciphertext $c \in \mathcal{C}$ that is the encryption of m using k. We denote this as $c = \mathit{Enc}(m, k)$.

- Dec is the *decryption algorithm*, a deterministic algorithm that accepts a key $k \in \mathcal{K}$, a ciphertext $c \in \mathcal{C}$, and yields a plaintext m that is the decryption of c using k. We denote this as $m = \mathit{Dec}(c, k)$.

- Decryption is the inverse of encryption. Formally, for all $m \in \mathcal{M}$ and $k \in \mathcal{K}$,

$$\mathsf{Pr}\left(\mathit{Dec}(\mathit{Enc}(m, k), k) = m\right) \;=\; 1.$$

A private-key encryption scheme satisfies *perfect indistinguishability* [32] if the probability distribution on ciphertexts is the same for all plaintexts. Formally, for all m_1, m_2, and c,

$$\mathsf{Pr}\left(k \leftarrow \mathit{Gen} : \mathit{Enc}(m_1, k) = c\right) \;=\; \mathsf{Pr}\left(k \leftarrow \mathit{Gen} : \mathit{Enc}(m_2, k) = c\right).$$

Perfect indistinguishability can be formulated as a hyperproperty on probabilistic systems. To encode encryption scheme ($\mathcal{M}$, $\mathcal{K}$, $\mathcal{C}$, *Gen*, *Enc*, *Dec*) as a probabilistic system, let the set of states of the system be

$$\mathcal{M} \cup \mathcal{K} \cup \mathcal{C} \cup \{\mathit{Gen}\} \cup \{\mathit{Enc}(m, k) \mid k \in \mathcal{K}, m \in \mathcal{M}\} \\ \cup \{\mathit{Dec}(c, k) \mid k \in \mathcal{K}, c \in \mathcal{C}\}.$$

Let probability function $p(\cdot, \cdot)$ be defined such that

- $p(\mathit{Gen}, k) \;=\; \mathsf{Pr}\left(k = \mathit{Gen}\right)$,

- $p(\mathit{Enc}(m, k), c) \;=\; \mathsf{Pr}\left(c = \mathit{Enc}(m, k)\right)$, and

- $p(\mathit{Dec}(c, k), m) = 1$ iff $\mathit{Dec}(c, k) = m$.

Let the system so constructed from $(\mathcal{M}, \mathcal{K}, \mathcal{C}, \mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ be denoted

$$\mathit{encSys}(\mathcal{M}, \mathcal{K}, \mathcal{C}, \mathit{Gen}, \mathit{Enc}, \mathit{Dec}),$$

and let the set of all such systems be ***ES***. The following hyperproperty expresses perfect indistinguishability:

$$\begin{aligned}\boldsymbol{PI} \triangleq \{T \in \mathsf{Prop} \mid T \in \boldsymbol{ES} \\ &\wedge (\exists\, \mathcal{M}, \mathcal{K}, \mathcal{C}, \mathit{Gen}, \mathit{Enc}, \mathit{Dec} : \\ &\quad T = \mathit{encSys}(\mathcal{M}, \mathcal{K}, \mathcal{C}, \mathit{Gen}, \mathit{Enc}, \mathit{Dec}) \\ &\quad\quad \wedge (\forall\, m_1, m_2 \in \mathcal{M}; c \in \mathcal{C} : \\ &\quad\quad\quad \mathsf{Pr}\,(\mathit{Enc}(m_1) = c) \\ &\quad\quad\quad\quad = \mathsf{Pr}\,(\mathit{Enc}(m_2) = c)))\},\end{aligned} \tag{7.6}$$

where $\mathsf{Pr}\,(\mathit{Enc}(m) = c)$ denotes

$$\sum_{k \in \mathcal{K}} \mathsf{Pr}_{\mathit{Gen},T}(\{\mathit{Gen}, k\}) \cdot \mathsf{Pr}_{\mathit{Enc}(m,k),T}(\{\mathit{Enc}(m,k), c\}).$$

PI is hypersafety for ***ES*** because any encryption scheme that is not in ***PI*** has a ciphertext c and two messages m_1, m_2 such that the probability that m_1 encrypts to c is not equal to the probability that m_2 encrypts to c. Trace set $\{\mathit{Enc}(m,k), c \mid k \in \mathcal{K}, m \in \{m_1, m_2\}\}$ thus is irremediable, and it is finite assuming that key space $\mathcal{K}$ is finite. So the trace set is a bad thing. But note that ***PI*** is not subset closed for Prop, so stepwise refinement is not applicable with ***PI***.

Other definitions of secure encryption, such as computational indistinguishability in various attacker models (including IND-CPA and IND-CCA), can similarly be formulated as hyperproperties.

Quantifying information flow. Probability can also be used to reason about the amount of information that a system can leak. For example, *channel capacity* is the maximum rate at which information can be reliably sent over a channel [55]; Gray [25] formulates as a channel the leakage of secret information from a system, and he quantifies the capacity of that channel. The hyperproperty "The channel capacity is k bits" (denoted $\boldsymbol{CC}_k$) is hyperliveness for ***PR***, since no matter what the rate is for some finite prefix of the system, the rate can changed to any arbitrary amount by an appropriate extension that conveys more or less information.

To measure *quantity of leakage from repeated experiments* in probabilistic programs, Clarkson et al. [14] use a probabilistic denotational semantics. This semantics can be used to define a system, and the traces of the system represent repeated executions of the program. The hyperproperty "The quantity of leakage over every series of experiments on program S is less than k bits" (denoted $\boldsymbol{QL}_k$) is hypersafety for a variant of ***PR***. For details, see Appendix B.

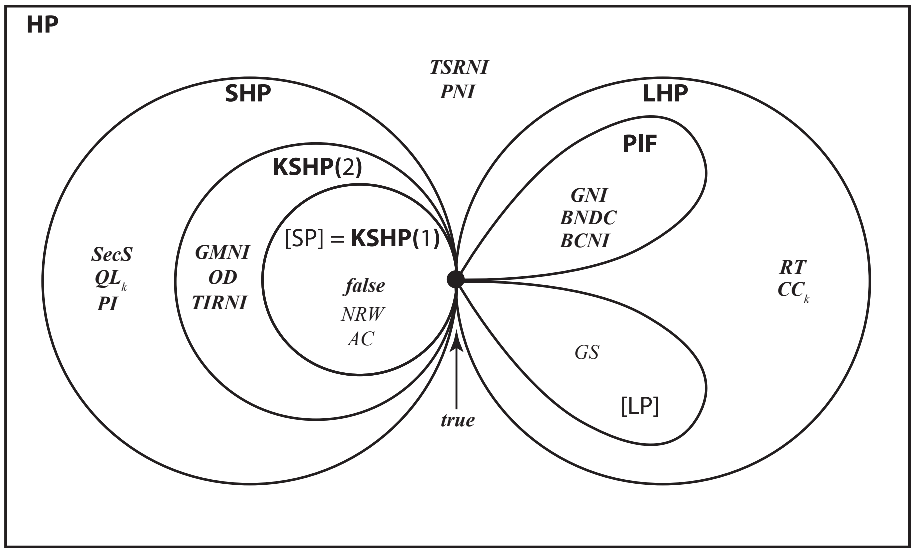

Figure 1: Classification of security policies

8 Concluding Remarks

Many security policies have been classified as hyperproperties in this paper. Figure 1 summarizes this classification.

Although this paper formulates security policies with hyperproperties, security policies historically have been formulated in terms of confidentiality, integrity, and availability requirements [16, 17, 31]. The relation between these two formulations is an open question, but we can offer some observations:

- Information-flow confidentiality is not a trace property, but it is a hyperproperty, and it can be hypersafety (e.g., observational determinism) or hyperliveness (e.g., generalized noninterference).
- Integrity, which we have not discussed in this paper, includes examples from safety, hypersafety, and hyperliveness.
- Availability is sometimes hypersafety (maximum response time in any execution, which is also safety) and sometimes hyperliveness (mean response time over all executions).

The classification of security requirements as confidentiality, integrity, and availability therefore would seem to be orthogonal to hypersafety and hyperliveness. Hypersafety and hyperliveness have the advantages of being formalized and providing an orthogonal basis for constructing security policies. In contrast, there is no formalization that simultaneously characterizes confidentiality, integrity, and availability,[32] nor are confidentiality, integrity, and availability orthogonal.[33]

Finally, no relatively complete verification methodology exists for confidentiality, integrity, or availability. But there is such a methodology for trace properties: given a trace property P, construct a safety property S and a liveness property L such that $P = S \cap L$, then use invariance arguments to verify S and well-foundedness arguments to verify L [4, 5]. And we have now taken steps toward generalizing this methodology to apply to hyperproperties. Theorem 5 shows that every hyperproperty $\boldsymbol{P}$ can be expressed as the intersection of a safety hyperproperty $\boldsymbol{S}$ and a liveness hyperproperty $\boldsymbol{L}$, and the proof of Theorem 5 shows that $\boldsymbol{S}$ and $\boldsymbol{L}$ can be constructed from $\boldsymbol{P}$. If $\boldsymbol{S}$ is a k-safety hyperproperty, then by Theorem 2, it can be verified using reasoning about safety. It remains an open question whether general methods exist that are relatively complete for verification of safety hyperproperties that are not k-safety, or for liveness hyperproperties.[34] Such methods would complete the verification methodology for hyperproperties. Then, security might take its place as "just another" functional requirement to be verified.

Acknowledgments

We thank Graeme Bailey, Stephen Chong, Dexter Kozen, Ueli Maurer, Andrew Myers, and Tom Roeder for discussions about this work. We also thank Martín Abadi, Borzoo Bonakdarpour, Stephen Chong, Michael George, Leslie Lamport, Jed Liu, John McLean, John Mitchell, Greg Morrisett, David Naumann, Tamara Rezk, Tom Roeder, and Tachio Terauchi for their comments on a draft of this paper.

References

[1] Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.

[2] Martín Abadi and Leslie Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, 1993.

[3] Samson Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51:1–77, 1991.

[4] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[32] The closest example of which we are aware is from Zheng and Myers [66], who formalize a particular noninterference policy for confidentiality, integrity, and availability.

[33] For example, the requirement that a principal be unable to read a value could be interpreted as confidentiality or unavailability of that value.

[34] If the full power of second-order logic is necessary to express hyperproperties (as discussed at the end of section 2), then such methods could not exist. Nonetheless, methods for verifying fragments of the logic might suffice for verifying hyperproperties that correspond to security policies.

[5] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[6] Ross J. Anderson. A security policy model for clinical information systems. In *Proc. of IEEE Symposium on Security and Privacy*, pages 30–43, Oakland, California, May 1996.

[7] Ralph-Johan R. Back. On correct refinement of programs. *Journal of Computer and System Sciences*, 23(1):49–68, August 1981.

[8] Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. Secure information flow by self-composition. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 100–114, Pacific Grove, California, June 2004.

[9] Johan van Benthem and Kees Doets. Higher-order logic. In *Elements of Classical Logic*, volume 1 of *Handbook of Philosophical Logic*, pages 275–330. D. Reidel Publishing, Dordrecht, Holland, 1983.

[10] Annalisa Bossi, Riccardo Focardi, Carla Piazza, and Sabina Rossi. Refinement operators and information flow security. In *IEEE Conference on Software Engineering and Formal Methods*, pages 44–53, Brisbane, Australia, June 2003.

[11] Gérard Boudol and Ilaria Castellani. Noninterference for concurrent programs and thread systems. *Theoretical Computer Science*, 281(1–2):109–130, 2002.

[12] Denis L. Bueno and Michael R. Clarkson. Hyperproperties: Verification of proofs. Technical report, Cornell University Computing and Information Science, July 2008. Available from `http://hdl.handle.net/1813/11153`.

[13] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[14] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Belief in information flow. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 31–45, Aix-en-Provence, France, June 2005.

[15] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *Proc. of IEEE Computer Security Foundations Symposium*, pages 51–65, Pittsburgh, Pennsylvania, June 2008.

[16] Commission of the European Communities (ECSC, EEC, EAEC). Information Technology Security Evaluation Criteria: Provisional harmonised criteria, June 1991. Document COM(90) 314, Version 1.2.

[17] Common Criteria for Information Technology Security Evaluation: Part 1: Introduction and general model, September 2005. CCMB-2006-09-001, Version 3.1, Revision 1. Available from `www.commoncriteriaportal.org`.

[18] Ole-Johan Dahl, C.A.R. Hoare, and Edsger W. Dijkstra. *Structured Programming*. Academic Press, London, 1972.

[19] Ádám Darvas, Reiner Hähnle, and David Sands. A theorem proving approach to analysis of secure information flow. In *Security in Pervasive Computing*, volume 3450 of *Lecture Notes in Computer Science*, pages 193–209. Springer, Berlin, 2005.

[20] Edsger W. Dijkstra. A constructive approach to the problem of program correctness. *BIT Numerical Mathematics*, 8:174–186, 1968.

[21] E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.

[22] Riccardo Focardi and Roberto Gorrieri. Classification of security properties (Part I: Information flow). In *Foundations of Security Analysis and Design 2000*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396. Springer, Berlin, 2001.

[23] Joseph A. Goguen and Jose Meseguer. Security policies and security models. In *Proc. of IEEE Symposium on Security and Privacy*, pages 11–20, Oakland, California, April 1982.

[24] James W. Gray, III. Probabilistic interference. In *Proc. of IEEE Symposium on Security and Privacy*, pages 170–179, Oakland, California, May 1990.

[25] James W. Gray, III. Towards a mathematical foundation for information flow security. In *Proc. of IEEE Symposium on Security and Privacy*, pages 21–34, Oakland, California, May 1991.

[26] Joseph Y. Halpern. *Reasoning About Uncertainty*. MIT Press, Cambridge, Massachusetts, 2003.

[27] Godfrey Harold Hardy. *Divergent Series*. Chelsea, New York, 1991.

[28] Jifeng He, C.A.R. Hoare, and Jeff W. Sanders. Data refinement refined. In *Proc. of European Symposium on Programming*, pages 187–196, Saarbrücken, Germany, March 1986.

[29] C.A.R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–281, 1972.

[30] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Englewood Cliffs, New Jersey, 1985.

[31] International Organization for Standardization. Information processing systems: Open systems interconnection—basic reference model. Part 2: Security architecture, 1989. ISO 7498-2.

[32] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, Boca Raton, Florida, 2008.

[33] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering.*, 3(2):125–143, 1977.

[34] Leslie Lamport. "Sometime" is sometimes "not never": On the temporal logic of programs. In *Proc. of Symposium on Principles of Programming Languages*, pages 174–185, Las Vegas, North Dakota, January 1980.

[35] Leslie Lamport. Basic concepts: Logical foundation. In *Distributed Systems: Methods and Tools for Specification, An Advanced Course*, volume 190 of *Lecture Notes in Computer Science*, pages 19–30. Springer, Berlin, 1985.

[36] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, Boston, 2002.

[37] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, New York, 1995.

[38] Heiko Mantel. Possibilistic definitions of security: An assembly kit. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 185–199, Cambridge, United Kingdom, July 2000.

[39] Heiko Mantel. Preserving information flow properties under refinement. In *Proc. of IEEE Symposium on Security and Privacy*, pages 78–91, Oakland, California, May 2001.

[40] Daryl McCullough. Specifications for multi-level security and a hook-up property. In *Proc. of IEEE Symposium on Security and Privacy*, pages 161–166, Oakland, California, April 1987.

[41] John McLean. Proving noninterference and functional correctness using traces. *Journal of Computer Security*, 1(1):37–58, 1992.

[42] John McLean. A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering.*, 22(1):53–67, 1996.

[43] Ernest Michael. Topologies on spaces of subsets. *Transactions of the American Mathematical Society*, 71(1):152–182, July 1951.

[44] Jonathan Millen. Covert channel capacity. In *Proc. of IEEE Symposium on Security and Privacy*, pages 60–66, Oakland, California, April 1987.

[45] Robin Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.

[46] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer-Verlag, Berlin, 2002.

[47] Kevin R. O'Neill, Michael R. Clarkson, and Stephen Chong. Information-flow security for interactive programs. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 190–201, Venice, Italy, July 2006.

[48] Gordon Plotkin. Domains. Unpublished manuscript of the Pisa notes on domain theory, 1983. Available from `http://homepages.inf.ed.ac.uk/gdp/publications/Domains.ps`.

[49] Francois Pottier and Vincent Simonet. Information flow inference for ML. In *Proc. of Symposium on Principles of Programming Languages*, pages 319–330, Portland, Oregon, January 2002.

[50] Riccardo Pucella and Fred B. Schneider. Independence from obfuscation: A semantic framework for diversity. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 230–241, Venice, Italy, July 2006.

[51] A. W. Roscoe. CSP and determinism in security modelling. In *Proc. of IEEE Symposium on Security and Privacy*, pages 114–127, Oakland, California, May 1995.

[52] John Rushby. Security requirements specifications: How and what? (extended abstract). Invited paper presented at *Symposium on Requirements Engineering for Information Security*, Indianapolis, Indiana, March 2001. Available from `http://www.csl.sri.com/users/rushby/abstracts/sreis01`.

[53] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, January 2003.

[54] Fred B. Schneider. *On Concurrent Programming*. Springer, New York, 1997.

[55] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[56] Stewart Shapiro. *Foundations Without Foundationalism: A Case for Second-Order Logic*. Clarendon Press, Oxford, 1991.

[57] Michael B. Smyth. Power domains and predicate transformers: A topological view. In *Proc. of International Colloquium on Automata, Languages, and Programming*, pages 662–675, Barcelona, Spain, July 1983.

[58] Michael B. Smyth. Topology. In *Background: Mathematical Structures*, volume 1 of *Handbook of Logic in Computer Science*, pages 641–762. Oxford University Press, 1992.

[59] David Sutherland. A model of information. In *National Security Conference*, pages 175–183, Gaithersburg, Maryland, 1986.

[60] Tachio Terauchi and Alexander Aiken. Secure information flow as a safety problem. In *Proc. of Static Analyses Symposium*, pages 352–367, London, United Kingdom, September 2005.

[61] Leopold Vietoris. Bereiche zweiter Ordnung. *Monatschefte für Mathematik und Physik*, 33:49–62, 1923.

[62] Dennis Volpano. Safety versus secrecy. In *Proc. of Static Analyses Symposium*, pages 303–311, Venice, Italy, September 1999.

[63] Niklaus Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221–227, April 1971.

[64] Aris Zakinthinos and E.S. Lee. A general theory of security properties. In *Proc. of IEEE Symposium on Security and Privacy*, pages 94–102, Oakland, California, May 1997.

[65] Steve Zdancewic and Andrew C. Myers. Observational determinism for concurrent program security. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 29–43, Pacific Grove, California, June 2003.

[66] Lantian Zheng and Andrew C. Myers. End-to-end availability policies and noninterference. In *Proc. of IEEE Computer Security Foundations Workshop*, pages 272–286, Aix-en-Provence, France, June 2005.

[67] Lidong Zhou, Fred B. Schneider, and Robbert van Renesse. APSS: Proactive secret sharing in asynchronous systems. *ACM Transactions on Information and System Security*, 8(3):259–286, August 2005.

A Summary of Notation

Bold face denotes "hyper" and $\mathsf{sans\ serif}$ denotes sets of (trace or hyper-) properties. Predicates and functions always begin with lower case, whereas (trace or hyper-) properties always begin with upper case.

Σ	set of all states
Ψ_{fin}	set of all finite traces
Ψ_{inf}	set of all infinite traces
Ψ	set of all traces
$t[i]$	trace index
$t[..i]$	trace prefix
$t[i..]$	trace suffix
Prop	set of all trace properties
$\mathcal{P}$	powerset operator
$\models$	trace property (and hyperproperty) satisfaction
NRW	trace property "no read then write"
AC	trace property "access control"
GS	trace property "guaranteed service"
$\mathsf{\mathbf{HP}}$	set of all hyperproperties
$[P]$	lift of trace property P to equivalent hyperproperty
$\boldsymbol{GMNI}$	hyperproperty "Goguen and Meseguer's noninterference"
$\boldsymbol{GNI}$	hyperproperty "generalized noninterference"
$\boldsymbol{OD}$	hyperproperty "observational determinism"
$=_L$	low-indistinguishability relation on states
$\approx_L$	low-indistinguishability relation on traces
$\boldsymbol{RT}$	hyperproperty "mean response time"
$\mathsf{\mathbf{SSC}}$	set of all subset-closed hyperproperties
SP	set of all safety properties
$\leq$	trace (or trace set) prefix
Obs	set of all observations
$\mathsf{\mathbf{SHP}}$	set of all safety hyperproperties
$\mathsf{\mathbf{KSHP}}(k)$	set of all k-safety hyperproperties
S^k	k-fold parallel self-composition
Sys	set of all systems
$\boldsymbol{SecS}$	hyperproperty "secret sharing"
LP	set of all liveness properties
$\mathsf{\mathbf{LHP}}$	set of all liveness hyperproperties
$\boldsymbol{true}$	hyperproperty that holds of all systems
$\boldsymbol{false}$	hyperproperty that holds of no systems
Cl	closure operator
$\mathsf{\mathbf{PIF}}$	set of all possibilistic information-flow hyperproperties
$\boldsymbol{NNT}$	hyperproperty "not never terminates"
SAG	trace property "sometime always good"
$\mathcal{O}$	open sets of Plotkin topology
$\uparrow$	completion of trace or observation

$\boldsymbol{\mathcal{O}}$	open sets of our topology
$\boldsymbol{\mathcal{C}}$	closed sets of our topology
$\boldsymbol{\mathcal{D}}$	dense sets of our topology
$\mathfrak{V}_L$	lower Vietoris construction
Cl_C	closure under infinite intersection and finite union
Rep	system representation
$Obs(\mathsf{Rep})$	observations of a system representation
Rel	system representation "relational systems"
TIRNI	hyperproperty "termination-insensitive noninterference"
TSRNI	hyperproperty "termination-sensitive noninterference"
LTS	system representation "labeled transition systems"
BNDC	hyperproperty "bisimulation nondeducibility on compositions"
SM	system representation "deterministic state machines"
PR	system representation "probabilistic systems"
$\mathsf{Pr}_{s,S}(T)$	probability that set T of finite traces is produced by probabilistic system S beginning in initial state s
PNI	hyperproperty "probabilistic noninterference"
ES	system representation "encryption schemes"
PI	hyperproperty "perfect indistinguishability"
$\boldsymbol{CC}_k$	hyperproperty "channel capacity"
$\boldsymbol{QL}_k$	hyperproperty "quantitative leakage"
BCNI	hyperproperty "Boudol and Castellani's noninterference"

B Longer Examples of Hyperproperties

B.1 Boudol and Castellani's Noninterference

Boudol and Castellani [11] define a bisimulation-based noninterference policy for concurrent programs. To model this policy as a hyperproperty, we first formalize their model of program execution. They model execution as a binary relation $\rightarrow$ on program terms and memories; a program term P and a memory μ step to a new program term P' and memory μ'. Define the set Σ_P of states for program P to be the set of pairs of a program term and a memory, $\mathit{prog}(s)$ to be the program term from state s, and $\mathit{mem}(s)$ to be the memory from state s. Define $\mathit{traces}(P)$ to be the set of all traces t such that $\mathit{prog}(t[0])$ is P, and for all i, $t[i] \rightarrow t[i+1]$. This construction encodes P as a set of traces and is an instance of our general construction for encoding LTSs (cf. section 7.2.2); here there are only LTS-states and no labels.

Second, we formalize Boudol and Castellani's security policy. Let $=_L$ be an equivalence relation on memories such that $\mu_1 =_L \mu_2$ means μ_1 and μ_2 are indistinguishable to a low observer. State s can *step* to state s' in program P, denoted $\mathit{steps}_P(s, s')$, if

$$(\exists\, t \in \Psi_{\mathsf{inf}}, i \in \mathbb{N} : t \in \mathit{traces}(P) \;\wedge\; t[i] = s \wedge t[i+1] = s').$$

Define $\approx_L^P$ (read "bisimilar") to be a binary relation on Σ_P such that if s_1 is bisimilar to s_2, then s_1 and s_2 must have indistinguishable memories to a low observer; further, if s_1 can step to state s_1', then either s_1' is bisimilar to s_2, or s_2 can step to s_2' where s_1' and s_2' are bisimilar. Formally, $\approx_L^P$ is the largest symmetric binary relation on Σ_P such that

$$\begin{aligned} s_1 \approx_L^P s_2 \implies\; & \mathit{mem}(s_1) =_L \mathit{mem}(s_2) \\ & \wedge\; (\exists\, s_1' \in \Sigma : \mathit{steps}_P(s_1, s_1') \\ & \qquad\qquad \implies s_1' \approx_L^P s_2 \\ & \qquad\qquad\qquad \vee\; (\exists\, s_2' \in \Sigma : \mathit{steps}_P(s_2, s_2') \\ & \qquad\qquad\qquad\qquad\qquad\qquad \wedge\; s_1' \approx_L^P s_2'))). \end{aligned}$$

Relation $\approx_L^P$ formalizes Definition 3.5 ($(\Gamma, \mathcal{L})$-Bisimulation) from [11].

Boudol and Castellani define program P to be secure, which we denote $\mathit{BCNI}(P)$, iff P is bisimilar to itself in all initially low-equivalent memories:

$$\mathit{BCNI}(P) \;\triangleq\; (\forall\, \mu_1, \mu_2 : \mu_1 =_L \mu_2 \implies (P, \mu_1) \approx_L^P (P, \mu_2)).$$

$\mathit{BCNI}(P)$ formalizes Definition 3.8 (Secure Programs) from [11]. The hyperproperty containing all secure programs according to Boudol and Castellani's definition is

$$\boldsymbol{\mathit{BCNI}} \;\triangleq\; \{T \in \mathsf{Prop} \mid T \in \boldsymbol{\mathit{LTS}} \implies (\exists\, P : T = \mathit{traces}(P) \;\wedge\; \mathit{BCNI}(P))\}.$$

BCNI is hyperliveness because of the existential quantifier on s_2' in the definition of $\approx_L^P$: any observation that contains traces leading to non-bisimilar states can be remedied by adding additional traces leading to bisimilar states. This remediation corresponds to a closure operator because it only adds traces, thus ***BCNI*** is a possibilistic information-flow policy.

B.2 Quantitative Information Flow

We summarize the model of Clarkson et al. [14]. A state has an immutable high component and a mutable low component. A *repeated experiment* on probabilistic program S is a finite sequence of executions of S. Each individual execution is an *experiment*. An execution is represented by two states: an initial state, in which inputs are provided to the program, and a final state, in which outputs are given by the program. All initial states (across all executions) in a repeated experiment must have the same high component but may have different low components. The probabilistic behavior of S is modeled by a semantics $[\![S]\!]$ that maps inputs states to output distributions, where $([\![S]\!]s)(s')$ is the probability that S begun in state s terminates in state s'. An attacker begins an experiment with a *prebelief* about the high component of the initial state. After observing the output of the execution, the attacker updates his prebelief to produce a *postbelief* about the high component of the initial state.

We here use traces and events to represent repeated experiments, where each state in a trace produces an event.[35] The events alternate between input and output, and the first event in a trace must be an input. Each output must have the correct probability of occurring according to $[\![S]\!]$ and the most recent input. Each low input component may vary, but the high component must be the same in each input. Let $Syst(S)$ denote the system of such traces resulting from program S:

$$\begin{aligned} Syst(S) \;\triangleq\; \{t \in \Psi_{\mathsf{fin}} \mid (\forall\, i : 0 \le 2i+1 \le |t| \\ \implies\; & ev_{Hin}(t[2i]) = ev_{Hin}(t[0]) \\ & \wedge\; p(t[2i], t[2i+1]) = ([\![S]\!]t[2i])(t[2i+1]))\}, \end{aligned}$$

where $|t|$ denotes the length of finite trace t, and $p(\cdot,\cdot)$ is the probability function used in section 7.2.4. From $Syst(S)$ we can construct probability measure $\mathsf{Pr}_{s,Syst(S)}$, also used in section 7.2.4.[36] The set of program states must be finite for the probability measure to be well-defined.

Each pair of states $t[i]$ and $t[i+1]$ (for even i) in repeated experiment t yields an experiment. An experiment is described formally by a prebelief, a high input, a low input, a low output, and a postbelief. As part of determining the postbelief for an experiment, the attacker's *prediction* δ_A of the low output is calculated from prebelief b_H and low input l:

$$\delta_A(b_H, l) \;\triangleq\; \lambda s \,.\, b_H(ev_{Hin}(s)) \cdot \mathsf{Pr}_{r,Syst(S)}(\{rs\}),$$

where r is the state that has $ev_{Hin}(s)$ as its high component and l as its low component. Denote the i^{th} experiment in trace t, with initial prebelief b_H, as $\mathcal{E}(t, i, b_H)$. We define

[35]A representation in which each finite trace contains two states (initial and final) might seem to be suitable for repeated experiments. That representation would fail to preserve the order in which inputs are provided (in initial states) across the sequence of executions in the repeated experiment. However, a single trace with many states does capture this order.

[36]Note that $p(s, s')$ is defined only at every other state in each trace of $Syst(S)$, so to construct the measure we treat each pair of states in the trace a single state.

$\mathcal{E}(t, i, b_H)$ using OCaml-style record syntax:

$$\begin{aligned}\mathcal{E}(t, i, b_H) \;\triangleq\; \{ & \; \mathit{preBelief} = \text{if } i > 0 \text{ then } \mathcal{E}(t, i-1).\mathit{postBelief} \text{ else } b_H;\\ & \mathit{highIn} = \mathit{ev}_{\mathit{Hin}}(t[2i]);\\ & \mathit{lowIn} = \mathit{ev}_L(t[2i]);\\ & \mathit{lowOut} = \mathit{ev}_L(t[2i+1]);\\ & \mathit{postBelief} = (\delta_A(b_H, l) \mid \mathit{lowOut}) \restriction H \;\},\end{aligned}$$

where $\mid$ is the distribution conditioning operator, and $\restriction$ is the distribution projection operator, from [14].

The quantity of flow in experiment $\mathcal{E}(t, i, b_H)$, denoted $\mathcal{Q}(\mathcal{E}(t, i, b_H))$, is defined in [14, §4]; we do not repeat the formalization here. The quantity of flow over repeated experiment t with initial prebelief b_H, denoted $\mathcal{Q}(t, b_H)$, is the sum of the flow for each experiment in t:

$$\mathcal{Q}(t, b_H) \;\triangleq\; \sum_{i=0}^{(|t|-1)/2} \mathcal{Q}(\mathcal{E}(t, i, b_H)).$$

Hyperproperty $\boldsymbol{QL}_k$ is the set of all systems that exhibit at most k bits of flow over any experiment:

$$\boldsymbol{QL}_k \;\triangleq\; \{T \in \mathsf{Prop} \mid (\exists\, S : T = \mathit{Syst}(S) \implies (\forall\, t \in T, b_H : \mathcal{Q}(b_H, t) \leq k))\}.$$

C System Representation Results

The results that appear before section 7.2 implicitly assume that the system representation is Prop. Section 7.2 generalizes those results to an arbitrary representation Rep, where Rep is a set of trace sets. We now give the formal statements of those generalized results.

Let $Tr(\mathsf{Rep})$ denote the set of all traces that are contained in any system in Rep—that is, $Tr(\mathsf{Rep}) = \bigcup_{T \in \mathsf{Rep}} T$. Let $Obs(Tr(\mathsf{Rep}))$ denote the set of all finite traces that are prefixes of some trace in $Tr(\mathsf{Rep})$—that is, $Obs(Tr(\mathsf{Rep})) = \{t \in \Psi_{\mathsf{fin}} \mid (\exists\, t' \in Tr(\mathsf{Rep}) : t \leq t')\}$. Let the lift $[P]_{\mathsf{Rep}}$ of property P in Rep be $\mathcal{P}(P) \cap \mathsf{Rep}$.

To generalize safety and liveness to system representations, it suffices to replace Ψ_{inf} with $Tr(\mathsf{Rep})$, and Ψ_{fin} with $Obs(Tr(\mathsf{Rep}))$. A trace property S is a *safety property for system representation* Rep iff

$$\begin{aligned}(\forall\, t \in Tr(\mathsf{Rep}) : t \notin S \implies (\exists\, m \in Obs(Tr(\mathsf{Rep})) : m \leq t \\ \wedge\ (\forall\, t' \in Tr(\mathsf{Rep}) : m \leq t' \implies t' \notin S))).\end{aligned}$$

A trace property L is a *liveness property for system representation* Rep iff

$$(\forall\, t \in Obs(Tr(\mathsf{Rep})) : (\exists\, t' \in Tr(\mathsf{Rep}) : t \leq t' \ \wedge\ t' \in L)).$$

Let $\mathsf{SP}(\mathsf{Rep})$ be the set of all safety properties for Rep, and let $\mathsf{LP}(\mathsf{Rep})$ be the set of all liveness properties for Rep. Likewise, let $\mathbf{SHP}(\mathsf{Rep})$ be the set of all safety hyperproperties for Rep, and let $\mathbf{LHP}(\mathsf{Rep})$ be the set of all liveness hyperproperties for Rep.

The following results are simple corollaries of the original results, although in some cases additional assumptions are needed about Rep.

Generalization of Proposition 1. If $(\forall\, t \in Tr(\mathsf{Rep}) : \{t\} \in \mathsf{Rep})$, then

$$(\forall\, S \in \mathcal{P}(\mathsf{Rep}) : S \in \mathsf{SP}(\mathsf{Rep}) \iff [S]_{\mathsf{Rep}} \in \mathbf{SHP}(\mathsf{Rep})).$$

The forward direction of this generalization always holds, but the backward direction ($\Longleftarrow$) might not hold if Rep does not allow individual traces from $Tr(\mathsf{Rep})$ to be representations: the bad thing for a safety hyperproperty could never be an individual trace, hence the safety hyperproperty could not be the lift of a safety property. So the backward direction requires the assumption that any individual trace in $Tr(\mathsf{Rep})$ is itself a system representation in Rep—that is, $(\forall\, t \in Tr(\mathsf{Rep}) : \{t\} \in \mathsf{Rep})$. Note that Prop satisfies this assumption.

Generalization of Proposition 2. If $(\forall\, T \subseteq Tr(\mathsf{Rep}) : T \in \mathsf{Rep})$, then

$$(\forall\, L \in \mathcal{P}(\mathsf{Rep}) : L \in \mathsf{LP}(\mathsf{Rep}) \iff [L]_{\mathsf{Rep}} \in \mathbf{LHP}(\mathsf{Rep})).$$

The backward direction of this generalization always holds, but the forward direction ($\Longrightarrow$) might not hold if Rep does not allow arbitrary unions of individual traces from $Tr(\mathsf{Rep})$ to be representations: the individual good things for a liveness property, when unioned, would not necessarily be good for the lift of that liveness property. So the forward direction requires the assumption that arbitrary unions of individual traces in $Tr(\mathsf{Rep})$ are themselves system representations in Rep—that is, $(\forall T \subseteq Tr(\mathsf{Rep}) : T \in \mathsf{Rep})$. Note that Prop satisfies this assumption.

Generalization of Theorem 1. If $(\exists L \in \mathsf{LP}(\mathsf{Rep}) : L \neq Tr(\mathsf{Rep}))$, then

$$\mathbf{SHP}(\mathsf{Rep}) \subset \mathbf{SSC}(\mathsf{Rep}).$$

$\mathsf{SSC}(\mathsf{Rep})$ is the set of all hyperproperties for Rep that are subset closed on Rep:

$$\boldsymbol{P} \in \mathbf{SSC}(\mathsf{Rep}) \iff (\forall T \in \boldsymbol{P} : (\forall T' \in \mathsf{Rep} : T' \subset T \implies T' \in \boldsymbol{P})).$$

The strictness of the subset in the theorem generalization requires the assumption that there exist subset-closed hyperproperties that are not safety. But it suffices to instead assume that hyperliveness is not trivial for Rep—that is, $(\exists L \in \mathsf{LP}(\mathsf{Rep}) : L \neq Tr(\mathsf{Rep}))$. Note that Prop satisfies both assumptions.

Generalization of Theorem 2.

$$(\forall S \in \mathsf{Rep}, \boldsymbol{K} \in \mathbf{KSHP}(k)(\mathsf{Rep}) : (\exists K \in \mathsf{SP}(\mathsf{Rep}) : S \models \boldsymbol{K} \iff S^k \models K)).$$

$\mathbf{KSHP}(k)(\mathsf{Rep})$ is the subset of $\mathbf{SHP}(\mathsf{Rep})$ where the size of bad thing M is bounded by k.

Generalization of Theorem 3. If there exists some liveness hyperproperty for Rep that is not a possibilistic information-flow policy for Rep, then

$$\mathbf{PIF}(\mathsf{Rep}) \subset \mathbf{LHP}(\mathsf{Rep}).$$

$\mathbf{PIF}(\mathsf{Rep})$ is the set of all possibilistic information-flow policies expressed by closure operators Cl of type $\mathsf{Rep} \to \mathsf{Rep}$. The strictness of the subset requires the assumption of the existence of a liveness hyperproperty for Rep that is not a possibilistic information-flow policy for Rep. Note that Prop satisfies this assumption.

Generalization of Theorem 5.

$$(\forall \boldsymbol{P} \in \mathcal{P}(\mathsf{Rep}) : (\exists \boldsymbol{S} \in \mathbf{SHP}(\mathsf{Rep}), \boldsymbol{L} \in \mathbf{LHP}(\mathsf{Rep}) : \boldsymbol{P} = \boldsymbol{S} \cap \boldsymbol{L})).$$

The proof of this generalization requires the following generalized definition:

$$\begin{aligned} \mathit{Safe}(\boldsymbol{P}) \;\triangleq\; \{T \in \mathsf{Rep} \mid (\forall M \in \mathit{Obs}(\mathsf{Rep}) : M \leq T \\ \implies (\exists T' \in \mathsf{Rep} : M \leq T' \wedge T' \in \boldsymbol{P}))\}. \end{aligned}$$

Also, in the definition of $\mathit{Live}(\boldsymbol{P})$, notation $\overline{\boldsymbol{H}}$ must now denote the complement of hyperproperty $\boldsymbol{H}$ with respect to Rep.

D Proofs

Bueno and Clarkson [12] have formally verified Propositions 1 and 2, Theorems 2, 3, and 5, and an analogue of Theorem 1 using the Isabelle/HOL proof assistant [46]. We believe that the remaining proofs could also be formally verified.

Proposition 1. $(\forall S \in \mathsf{Prop} : S \in \mathsf{SP} \iff [S] \in \mathsf{SHP})$.

Proof. By mutual implication.

($\Rightarrow$) Let S be an arbitrary safety property. We want to show that $[S]$ is a safety hyperproperty—that is, any trace property T not in $[S]$ contains some bad thing.

First, we find a bad thing M for T. By the definition of lifting, $[S] = \mathcal{P}(S) = \{P \in \mathsf{Prop} \mid P \subseteq S\}$. Since T is not in this set, $T \not\subseteq S$. So some trace t is in T but not in S. By the definition of safety, if $t \notin S$, there is some finite trace m that is a bad thing for S. So no extension of m is in S. Define M to be $\{m\}$.

Second, we show that M is irremediable. Note that $M \leq T$ because $m \leq t$ and $t \in T$. Let T' be an arbitrary trace property that extends M—that is, $M \leq T'$. By the definition of $\leq$, there exists a $t' \in T'$ such that $m \leq t'$. We established above that no extension of m is in S, so $t' \notin S$. But, again by the definition of lifting, $T' \notin [S]$, since T' contains a trace not in S.

Thus, by definition, $[S]$ is hypersafety.

($\Leftarrow$) Let S be an arbitrary trace property such that $[S]$ is hypersafety. We want to show that S is safety. Our strategy is as above—we find a bad thing and then show that it is irremediable.

Consider any t such that $t \notin S$. By the definition of lifting, we have that $\{t\} \notin [S]$. By the definition of hypersafety applied to $[S]$, there exists an $M \leq \{t\}$ such that for all $T' \geq M$, we have $T' \notin [S]$.

We claim that M must be non-empty. To show this, suppose for sake of contradiction that M is empty. Then M is a prefix of every trace property T', so no T' can be a member of S, which implies that $[S]$ itself must be empty. But $[S] = \mathcal{P}(S)$, so $[S]$ must at least contain S as a member. This is a contradiction, thus M is non-empty and contains at least one trace.

All traces in M must be prefixes of t, by the definition of $\leq$. Choose the longest such prefix in M and denote it as m^*. This m^* serves as a bad thing for t, as we show next.

Let t' be arbitrary such that $m^* \leq t'$, and let $T' = \{t'\}$. By the transitivity of $\leq$, we have $M \leq T'$, so $T' \notin [S]$ by the above application of the definition of hypersafety. But this implies that $t' \notin S$, by the definition of lifting.

We have shown that, for any $t \notin S$, there exists an $m \leq t$, such that for any $t' \geq m$, we have $t' \notin S$. Therefore, S is safety, by definition. □

Theorem 1. $\mathsf{SHP} \subset \mathsf{SSC}$.

Proof. Assume that $\boldsymbol{S}$ is hypersafety. For sake of contradiction, also assume that $\boldsymbol{S}$ is not subset closed. This latter assumption implies that there exist two trace properties T and T' such that $T \in \boldsymbol{S}$, and $T' \notin \boldsymbol{S}$, yet $T' \subset T$. By the definition of hypersafety, since $T' \notin \boldsymbol{S}$, there exists an observation M that is a bad thing for T'—that is, $M \leq T'$ and for all T'' such that $M \leq T''$, it holds that $T'' \notin \boldsymbol{S}$. Consider this M. By the definition of $\leq$, since $T' \subset T$ and $M \leq T'$, we have $M \leq T$. Then T is an instance of T'' above, which means $T \notin \boldsymbol{S}$. But this contradicts $T \in \boldsymbol{S}$. Therefore, $\boldsymbol{S}$ must be subset closed.

To see that the subset relation is strict, define the trace property $true$ as Ψ_{inf}. Consider any liveness property L other than $true$—for example, guaranteed service GS (2.3). When lifted to hyperproperty $[L]$, the result is subset closed by definition of $[\cdot]$. By Proposition 2 below (whose proof does not depend on this theorem), $[L]$ is hyperliveness. Since L is not $true$, we have that $[L]$ is not ***true***, which is the only hyperproperty that is both hypersafety and hyperliveness. So $[L]$ cannot be hypersafety. Thus $[L]$ is a hyperproperty that is not hypersafety but is subset closed. ☐

Theorem 2. $(\forall\, S \in \mathsf{Sys}, \boldsymbol{K} \in \mathsf{KSHP}(k) : (\exists\, K \in \mathsf{SP} : S \models \boldsymbol{K} \iff S^k \models K))$.

Proof. Let $\boldsymbol{K}$ be an arbitrary k-safety hyperproperty of system S. Our strategy is to construct a safety property K that holds of system S^k exactly when $\boldsymbol{K}$ holds of S.

Since $\boldsymbol{K}$ is k-safety, every trace property not contained in it has some bad thing of size at most k—that is, for all $T \notin \boldsymbol{K}$, there exists an observation M where $|M| \leq k$ and $M \leq T$, such that for all T' where $M \leq T'$, it holds that $T' \notin \boldsymbol{K}$. Construct the set $\boldsymbol{M}$ of all such bad things:

$$\begin{aligned}\boldsymbol{M} \;\triangleq\; \{M \in \mathsf{Obs} \mid |M| \leq k \,\wedge\, &(\exists\, T \in \mathsf{Prop} : T \notin \boldsymbol{K} \,\wedge\, M \leq T)\\ &\wedge\, (\forall\, T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \boldsymbol{K})\}.\end{aligned}$$

Next we define some notation to encode a set of traces as a single trace. Consider a trace property T such that $|T| \leq k$. Construct a finite list of traces $t_1, t_2, \ldots, t_k$ such that $t_i \in T$ for all i. Further, we require that no t_i is equal to any t_l, for any i and l, unless $|T| < k$. We construct a trace t such that $t[j]$ is the tuple $(t_1[j], t_2[j], \ldots, t_k[j])$; note that t is a trace over state space Σ^k. Let trace t so constructed from T be denoted $zip_k(T)$, and let the inverse of this construction be denoted $unzip_k(t)$; note that $zip_k(\cdot)$ and $unzip_k(\cdot)$ are partial functions. We can also apply this notation to observations, which are finite sets of finite traces.[37]

Now we can construct safety property K. Let K be the set of traces over Σ^k such that no trace in K encodes an extension of any bad thing $M \in \boldsymbol{M}$:

$$K \;\triangleq\; \{t^k \mid \neg(\exists\, M \in \mathsf{Obs} : M \in \boldsymbol{M} \,\wedge\, zip_k(M) \leq t^k)\},$$

[37]In this case, the t_i have finite and potentially differing length. So if $j > |t_i|$, let $t_i[j] = \bot$ for some new state $\bot \notin \Sigma$. Thus, $zip_k(T)$ is a trace over state space $(\Sigma \cup \bot)^k$. We redefine trace prefix $\leq$ over this space to ignore $\bot$: let $t \leq t'$ iff, for some t'' that is a trace over Σ, $\lceil t \rceil = \lceil t' \rceil t''$, where $\lceil t \rceil$ is the truncation of t that removes any $\bot$ states. For notational simplicity, we omit this technicality in the remainder of the proof.

where t^k denotes a trace t over space Σ^k.

To see that K is safety, suppose that $t^k \notin K$. Then by the definition of K, there must exist some $M \in \boldsymbol{M}$ such that $zip_k(M) \leq t^k$. Consider any trace $u^k \geq zip_k(M)$. By the definition of K, we have that $u^k \notin K$. Thus, for any trace t^k not in K, there is some finite bad thing $zip_k(M)$, such that no extension u^k of the bad thing is in K. By definition, K is therefore safety.

Finally, we need to show that S satisfies $\boldsymbol{K}$ exactly when S^k satisfies K. We do so by mutual implication.

($\Rightarrow$) Suppose $S \models \boldsymbol{K}$. Then, by definition, $S \in \boldsymbol{K}$. For sake of contradiction, suppose that $S^k \not\subseteq K$. Then, by the definition of subset, there exists some $t^k \in S^k$ such that $t^k \notin K$. Let T be $unzip_k(t^k)$. By the definition of K, there must exist some $M \in \boldsymbol{M}$ such that $zip_k(M) \leq t^k$. Applying $unzip_k(\cdot)$ to this predicate, and noting that $unzip$ is monotonic with respect to $\leq$, we obtain $M \leq unzip_k(t^k)$. By the definition of T, we then have that $M \leq T$. By the construction of $\boldsymbol{M}$, T therefore cannot be in $\boldsymbol{K}$. By the construction of S^k and the definition of T, each trace in T must also be a trace of S. So by definition, $T \leq S$. By transitivity, we have that $M \leq S$. By the construction of $\boldsymbol{M}$, S then cannot be in $\boldsymbol{K}$. But this contradicts the fact that $S \in \boldsymbol{K}$. Therefore, $S^k \subseteq K$, so by definition $S^k \models K$.

($\Leftarrow$) Suppose $S^k \models K$. Then, by definition, $S^k \subseteq K$. Suppose, for sake of contradiction, that S does not satisfy $\boldsymbol{K}$. Then, by definition, $S \notin \boldsymbol{K}$. Since $\boldsymbol{K}$ is k-safety, this means that there exists an $M \leq S$, where $|M| \leq k$, such that for all $T' \geq M$, $T' \notin \boldsymbol{K}$. Let m^k be $zip_k(M)$, and let s^k be a trace of S^k such that $m^k \leq s^k$ (such a trace must exist since $M \leq S$). By the construction of K, for any $t^k \geq m^k$, we have that $t^k \notin K$. Therefore, $s^k \notin K$, and it follows that $S^k \not\subseteq K$. But this contradicts the fact that $S^k \subseteq K$. Therefore, $S \in \boldsymbol{K}$, so by definition $S \models \boldsymbol{K}$. $\square$

Proposition 2. $(\forall\, L \in \mathsf{Prop} : L \in \mathsf{LP} \iff [L] \in \mathbf{LHP})$.

Proof. By mutual implication.

($\Rightarrow$) Let L be an arbitrary liveness property. We want to show that $[L]$ is a liveness hyperproperty—that is, any observation M can be extended to a trace property T that is contained in $[L]$. So let M be an arbitrary observation. By the definition of liveness, for each $m \in M$, there exists some $t \geq m$ such that $t \in L$. For a given m, let that trace t be denoted t_m. Construct the set $T = \bigcup_{m \in M}\{t_m\}$. Since all the t_m are elements of L, we have $T \subseteq L$. By the definition of lifting, it follows that T is contained in $[L]$. Further, T extends M by the construction of T. Thus, T satisfies the requirements of the trace property we needed to construct. By definition, $[L]$ is hyperliveness.

($\Leftarrow$) Let L be an arbitrary property such that $[L]$ is hyperliveness. We want to show that L is liveness. So consider an arbitrary trace t, and let $T = \{t\}$. Since $[L]$ is hyperliveness, we have that there exists a T' such that $T \leq T'$ and $T' \in [L]$. Since $T \leq T'$ and $T = \{t\}$, there exists a t' such that $t \leq t'$ and $t' \in T'$, by the definition of $\leq$. By the definition of lifting, if $t' \in T' \in [L]$, then it must be the case that $t' \in L$. Thus, for any t, there exists a t' such that $t \leq t'$ and $t' \in L$. Therefore, L is liveness, by definition. $\square$

Theorem 3. $\mathsf{PIF} \subset \mathsf{LHP}$.

Proof. Let $\boldsymbol{P}$ be an arbitrary possibilistic information-flow hyperproperty, and let $Cl_{\boldsymbol{P}}$ be the closure operator that Mantel [38] would associate with $\boldsymbol{P}$.[38] Then, by Mantel's Definition 10, it must be the case that $\boldsymbol{P} = \{Cl_{\boldsymbol{P}}(T) \mid T \in \mathsf{Prop}\}$. Closure operators must satisfy the axiom $(\forall X : X \subseteq Cl(X))$, which we use below.

To show that $\boldsymbol{P}$ is hyperliveness, let $T \in \mathsf{Obs}$ be arbitrary. By the definition of hyperliveness, we need to show that there exists a $T' \in \mathsf{Prop}$ such that $T \leq T'$ and $T' \in \boldsymbol{P}$. Let T' be $Cl_{\boldsymbol{P}}(\hat{T})$, where $\hat{T}$ denotes the embedding of T into Prop by infinitely stuttering the final state of each trace in T, as discussed in section 2. By the closure axiom above, we have that $\hat{T} \subseteq Cl_{\boldsymbol{P}}(\hat{T})$. So by the definition of $\leq$, we can conclude $T \leq Cl_{\boldsymbol{P}}(\hat{T}) = T'$. Further, T' must be an element of $\boldsymbol{P}$ since it is the $Cl_{\boldsymbol{P}}$-closure of trace property $\hat{T}$. Therefore, T' satisfies the required conditions, and $\boldsymbol{P}$ is hyperliveness.

To see that the subset relation is strict, consider liveness property GS (guaranteed service) from section 2. It corresponds to liveness hyperproperty $[GS]$, but has no corresponding closure operator. For suppose that such a closure operator did exist, and consider an infinite trace t in which service fails to occur. The closure of any set containing t must still contain t, by the axiom above. But then the closure does not satisfy GS, and so the closure operator cannot correspond to $[GS]$. $\square$

Proposition 3. $\boldsymbol{\mathcal{O}}^{B} = \boldsymbol{\mathcal{O}}^{SB}$.

Proof. By mutual containment.

($\supseteq$) By definition, the elements of $\boldsymbol{\mathcal{O}}^{B}$ are finite intersections of elements of $\boldsymbol{\mathcal{O}}^{SB}$. Thus, every element of $\boldsymbol{\mathcal{O}}^{SB}$ is already trivially an element of $\boldsymbol{\mathcal{O}}^{B}$.

($\subseteq$) Let $\boldsymbol{N}$ be an arbitrary element of $\boldsymbol{\mathcal{O}}^{B}$. By the definition of a base, we can write $\boldsymbol{N}$ as $\bigcap_i \uparrow M_i$, where i ranges over a finite index set and each M_i is an observation. We want to show that there exists an element $\uparrow N$ of $\boldsymbol{\mathcal{O}}^{SB}$ such that $\boldsymbol{N} = \uparrow N$. So consider $\boldsymbol{N}$. Every trace property T in it must extend every M_i. Thus, by the definition of $\leq$, every such trace property T extends $\bigcup_i M_i$. Therefore $\boldsymbol{N} = \uparrow \bigcup_i M_i$. Our desired observation N is thus $\bigcup_i M_i$. Note that, for N to be a valid observation, it must be a finite set. The union over M_i must therefore result in a finite set—which it does, since i ranges over a finite index set. $\square$

[38]More precisely, Mantel argues that every "possibilistic information-flow property [*sic*]" can be expressed as a *basic security predicate*, and that each basic security predicate induces a set of closure operators. Any element of this set suffices to instantiate $Cl_{\boldsymbol{P}}$. Also, Mantel's closure operators were over finite traces, and we have generalized to infinite traces.

Proposition 4. $\mathsf{SHP} = \mathcal{C}$.

Proof. By mutual containment.

($\subseteq$) Let $\boldsymbol{S}$ be an arbitrary safety hyperproperty. We need to show that it is also a closed set. By the definition of closed, this is equivalent to showing that $\boldsymbol{S}$ is the complement of an open set. Our strategy is to construct hyperproperty $\boldsymbol{O}$, show that $\overline{\boldsymbol{O}}$ and $\boldsymbol{S}$ are equal, and show that $\boldsymbol{O}$ is open.

By the definition of hypersafety, we have that any trace property T that is not a member of $\boldsymbol{S}$—and thus is a member of $\overline{\boldsymbol{S}}$—must contain some bad thing. Consider the set $\boldsymbol{M} \in \mathcal{P}(\mathsf{Obs})$ of all bad things for $\boldsymbol{S}$. $\boldsymbol{M}$ contains one or more elements for every trace property in $\overline{\boldsymbol{S}}$:

$$\begin{aligned} \boldsymbol{M} \triangleq \{M \in \mathsf{Obs} \mid (\exists\, T \in \overline{\boldsymbol{S}} : M \leq T \\ \wedge\ (\forall\, T' \in \mathsf{Prop} : M \leq T' \implies T' \in \overline{\boldsymbol{S}}))\}. \end{aligned}$$

Next, define $\boldsymbol{O}$ as the completion of $\boldsymbol{M}$—that is, the set of all trace properties that extend a bad thing for $\boldsymbol{S}$:

$$\begin{aligned} \boldsymbol{O} &\triangleq \bigcup_{M \in \boldsymbol{M}} \uparrow M \\ &= \{T \mid (\exists\, M \in \boldsymbol{M} : M \leq T)\}, \end{aligned} \tag{D.1}$$

where the equality follows by the definition of $\uparrow M$. Since each such trace property T violates $\boldsymbol{S}$, we would suspect that $\boldsymbol{O}$ is the complement of $\boldsymbol{S}$. This is indeed the case:

Claim. $\boldsymbol{O} = \overline{\boldsymbol{S}}$

Proof. By mutual containment.

($\subseteq$) Suppose $T \in \boldsymbol{O}$. Then by equation D.1, there is some $M \in \boldsymbol{M}$ such that $M \leq T$. By the definition of $\boldsymbol{M}$, any extension of M is an element of $\overline{\boldsymbol{S}}$. Since T is such an extension, $T \in \overline{\boldsymbol{S}}$.

($\supseteq$) Suppose $T \in \overline{\boldsymbol{S}}$. Then $T \notin \boldsymbol{S}$, so by the definition of hypersafety, $(\exists\, M \in \mathsf{Obs} : M \leq T \ \wedge\ (\forall\, T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \boldsymbol{S}))$. Consider that M. It must be a member of $\boldsymbol{M}$, by definition. Since $M \leq T$, we have that $T \in \boldsymbol{O}$ by equation D.1. □

All that remains is to show that $\boldsymbol{O}$ is open. First, note that $\uparrow M$, for any $M \in \mathsf{Obs}$, is by definition an element of $\mathcal{O}^{SB}$. Thus each of the sets $\uparrow M$ in the definition of $\boldsymbol{O}$ is open. Second, by the definition of open sets, a union of open sets is open. $\boldsymbol{O}$ is such a union, and is therefore open.

($\supseteq$) Let $\boldsymbol{C}$ be an arbitrary closed set. We need to show that it is also hypersafety. Our strategy is to identify, for any trace property T not in $\boldsymbol{C}$, a bad thing for T. If such a bad thing exists for all T, then $\boldsymbol{C}$ is by definition hypersafety.

Since $\boldsymbol{C}$ is closed, it is by definition the complement of an open set. By Proposition 3, we can therefore write $\overline{\boldsymbol{C}}$ as follows:

$$\overline{\boldsymbol{C}} = \bigcup_i \uparrow M_i, \tag{D.2}$$

where each M_i is an observation.

Let T be an arbitrary trace property such that $T \notin \boldsymbol{C}$, or equivalently, such that $T \in \overline{\boldsymbol{C}}$. Then T must be in at least one of the infinite unions in equation D.2. Thus, there must exist an i such that

$$T \in \uparrow M_i \qquad \text{and} \qquad M_i = \{U \in \mathsf{Prop} \mid M_i \leq U\}, \tag{D.3}$$

where the equality follows from the definition of $\uparrow$.

We construct the bad thing M for T by defining:

$$M \triangleq M_i.$$

We have that $M \leq T$, because of equation D.3.

To show that M is a bad thing for T, consider any $T' \geq M$. By the definition of M, $T' \geq M_i$. By equation D.3, it follows that T', like T, is a member of $\uparrow M_i$. By equation D.2, $T' \in \overline{\boldsymbol{C}}$. Therefore, $T' \notin \boldsymbol{C}$.

We have now shown that for any $T \notin \boldsymbol{C}$, there exists an $M \leq T$, such that for all $T' \geq M$, $T' \notin \boldsymbol{C}$. Thus $\boldsymbol{C}$ is hypersafety, by definition. ☐

Proposition 5. $\mathsf{LHP} = \boldsymbol{\mathcal{D}}$.

Proof. By mutual containment.

($\subseteq$) Let $\boldsymbol{L}$ be an arbitrary liveness hyperproperty. We need to show that $\boldsymbol{L}$ is dense. By the definition of dense, we must therefore show that $\boldsymbol{L}$ intersects every non-empty open set. So let $\boldsymbol{O}$ be an arbitrary non-empty open set. We need to show that $\boldsymbol{L} \cap \boldsymbol{O}$ is non-empty. By Proposition 3 and the definition of open, we can write $\boldsymbol{O}$ as $\bigcup_i \uparrow M_i$. Consider an arbitrary M_i. Since $\boldsymbol{L}$ is hyperliveness, there exists a $T \geq M_i$ such that $T \in \boldsymbol{L}$. Further, by the definition of $\uparrow$, we have that $T \in \boldsymbol{O}$. Therefore, $T \in \boldsymbol{L} \cap \boldsymbol{O}$, and it follows that $\boldsymbol{L}$ is dense, by definition.

($\supseteq$) Let $\boldsymbol{D}$ be an arbitrary dense set. To show that $\boldsymbol{D}$ is hyperliveness, we must show that any observation T can be extended to a trace property T' contained in $\boldsymbol{D}$—that is, $(\forall T \in \mathsf{Obs} : (\exists T' \in \mathsf{Prop} : T \leq T' \wedge T' \in \boldsymbol{D}))$. So let T be an arbitrary observation. Let $\boldsymbol{O}_T$ be the completion of T:

$$\begin{aligned} \boldsymbol{O}_T &\triangleq \uparrow T \\ &= \{T' \in \mathsf{Prop} \mid T \leq T'\} \end{aligned} \tag{D.4}$$

$\boldsymbol{O}_T$ is an element of $\boldsymbol{\mathcal{O}}^{SB}$, the subbase of our topology, by definition. Thus, by the definition of a subbase, $\boldsymbol{O}_T$ is an open set. By the definition of a dense set (which is that a dense set intersects every open set), we therefore have that $\boldsymbol{O}_T \cap \boldsymbol{D} \neq \emptyset$. Let T' be any element in the set $\boldsymbol{O}_T \cap \boldsymbol{D}$. By equation D.4, we have $T \leq T'$.

We have now shown that, for an arbitrary observation T, there exists a trace property T' such that $T \leq T'$ and $T' \in \boldsymbol{D}$. Therefore, $\boldsymbol{D}$ is hyperliveness, by definition. $\square$

Theorem 4. $\boldsymbol{\mathcal{O}} = \mathfrak{V}_L(\mathcal{O})$.

Proof. By mutual containment.

($\subseteq$) Suppose $\boldsymbol{O} \in \boldsymbol{\mathcal{O}}$. By the definitions of a base and of $\boldsymbol{\mathcal{O}}$, we can write $\boldsymbol{O}$ as $\bigcup_i^\infty \uparrow M_i$, where each M_i is an element of Obs.[39] Now we calculate:

$$\begin{aligned}
& \textstyle\bigcup_i^\infty \uparrow M_i \\
= & \quad \langle \text{ definition of } \uparrow \rangle \\
& \textstyle\bigcup_i^\infty \{T \mid T \geq M_i\} \\
= & \quad \langle \text{ definition of } \leq \ \rangle \\
& \textstyle\bigcup_i^\infty \{T \mid (\forall^* m_{ij} \in M_i \, : \, (\exists t \in T \, : \, m_{ij} \leq t))\} \\
= & \quad \langle \text{ definition of } \uparrow \rangle \\
& \textstyle\bigcup_i^\infty \{T \mid (\forall^* m_{ij} \in M_i \, : \uparrow m_{ij} \cap T \neq \emptyset)\} \\
= & \quad \langle \text{ definition of } \langle\cdot\rangle \ \rangle \\
& \textstyle\bigcup_i^\infty \{T \mid (\forall^* m_{ij} \in M_i \, : \, T \in \langle \uparrow m_{ij} \rangle)\} \\
= & \quad \langle \text{ definition of } \cap \ \rangle \\
& \textstyle\bigcup_i^\infty \bigcap_j^* \langle \uparrow m_{ij} \rangle
\end{aligned}$$

Since $\uparrow m_{ij} \in \mathcal{O}^B$ by definition, and $\mathcal{O}^B \subseteq \mathcal{O}$ by the definition of base, we have that $\langle \uparrow m_{ij} \rangle \in \mathfrak{V}_L^{SB}(\mathcal{O})$. Thus, by the definition of subbase, $\bigcup_i^\infty \bigcap_j^* \langle \uparrow m_{ij} \rangle \in \mathfrak{V}_L(\mathcal{O})$. Therefore, by the calculation above, we can conclude $\boldsymbol{O} \in \mathfrak{V}_L(\mathcal{O})$.

($\supseteq$) Suppose $\boldsymbol{O} \in \mathfrak{V}_L(\mathcal{O})$. By the definition of subbase and $\mathfrak{V}_L$, we can write $\boldsymbol{O}$ as $\bigcup_i^\infty \bigcap_j^* \langle O_{ij} \rangle$, where each O_{ij} is an element of $\mathcal{O}$. Now we calculate:

$$\begin{aligned}
& \textstyle\bigcup_i^\infty \bigcap_j^* \langle O_{ij} \rangle \\
= & \quad \langle \text{ definition of } \langle\cdot\rangle \ \rangle \\
& \textstyle\bigcup_i^\infty \bigcap_j^* \{T \mid T \cap O_{ij} \neq \emptyset\}
\end{aligned}$$

Since O_{ij} is open in the base topology $\mathcal{O}$, it can be rewritten a union of base

[39]We decorate quantifiers with ∞ and $*$ to denote an infinite and finite range, respectively.

open sets $\uparrow t_{ijk}$, where each t_{ijk} is a finite trace:

$$O_{ij} \;=\; \bigcup_{k}^{\infty} \uparrow t_{ijk}.$$

We continue calculating:

$=$ $\quad\langle$ rewriting O_{ij} $\rangle$

$\quad\bigcup_i^\infty \bigcap_j^* \{T \mid T \cap (\bigcup_k^\infty \uparrow t_{ijk}) \neq \emptyset\}$

$=$ $\quad\langle$ set theory $\rangle$

$\quad\bigcup_i^\infty \{T \mid (\forall^* j : (\exists^\infty k : T \cap \uparrow t_{ijk} \neq \emptyset))\}$

$=$ $\quad\langle$ definition of $\leq$ $\rangle$

$\quad\bigcup_i^\infty \{T \mid (\forall^* j : (\exists^\infty k : \{t_{ijk}\} \leq T))\}$

$=$ $\quad\langle$ set theory; let k' be the k guaranteed to exist for i and j $\rangle$

$\quad\bigcup_i^\infty \{T \mid \bigcup_j^* t_{ijk'} \leq T\}$

$=$ $\quad\langle$ let $M_i = \bigcup_j^* t_{ijk'}$ $\rangle$

$\quad\bigcup_i^\infty \{T \mid M_i \leq T\}$

$=$ $\quad\langle$ definition of $\uparrow$ $\rangle$

$\quad\bigcup_i^\infty \uparrow M_i$

Finally, since M_i is a finite set of finite traces, it is an element of Obs. So by definition, $\uparrow M_i \in \boldsymbol{\mathcal{O}}^{SB}$. Thus by the definition of base, $\bigcup_i^\infty \uparrow M_i \in \boldsymbol{\mathcal{O}}$. Therefore, by the calculation above, we can conclude $\boldsymbol{O} \in \boldsymbol{\mathcal{O}}$. □

Proposition 6. $\mathsf{SHP} = Cl_C(\{[S] \mid S \in \mathsf{SP}\})$.

Proof. Let ***S*** be an arbitrary safety hyperproperty. By Proposition 4, ***S*** is a closed set in topology $\boldsymbol{\mathcal{O}}$. By Theorem 4, ***S*** is thus also a closed set in topology $\mathfrak{V}_L(\mathcal{O})$. By the definition of closed, ***S*** is the complement of an open set in topology $\mathfrak{V}_L(\mathcal{O})$. By the definition of a base, we can thus write $\overline{\boldsymbol{S}}$ as unions of intersections of base elements. Letting $\sim$ denote set complement, we calculate:

$\quad\overline{\boldsymbol{S}}$

$=$ $\quad\langle$ definition of base $\rangle$

$\quad\bigcup_i^\infty \bigcap_j^* \langle O_{ij} \rangle$

$=$ $\langle$ definition of $\langle\cdot\rangle$ $\rangle$

$\bigcup_i^\infty \bigcap_j^* \{T \mid T \cap O_{ij} \neq \emptyset\}$

$=$ $\langle$ double negation $\rangle$

$\sim\sim\bigcup_i^\infty \bigcap_j^* \{T \mid T \cap O_{ij} \neq \emptyset\}$

$=$ $\langle$ set theory $\rangle$

$\sim\bigcap_i^\infty \bigcup_j^* \{T \mid T \cap O_{ij} = \emptyset\}$

$=$ $\langle$ set theory $\rangle$

$\sim\bigcap_i^\infty \bigcup_j^* \{T \mid T \subseteq \overline{O_{ij}}\}$

$=$ $\langle$ definition of $[\cdot]$ $\rangle$

$\sim\bigcap_i^\infty \bigcup_j^* [\overline{O_{ij}}]$

Removing a complement from each side of the above equation, we obtain

$$\boldsymbol{S} = \bigcap_i^\infty \bigcup_j^* [\overline{O_{ij}}].$$

Since each O_{ij} is open in topology $\mathcal{O}$, we have that $\overline{O_{ij}}$ is closed in $\mathcal{O}$. By the fact that closed sets in $\mathcal{O}$ correspond to safety properties [4], $\overline{O_{ij}}$ is a safety property. Therefore, $\boldsymbol{S}$ is the infinite intersection of finite unions of safety properties, and by definition of Cl_C must be an element of $Cl_C(\{[S] \mid S \in \mathsf{SP}\})$.

Similarly, given an arbitrary element of $Cl_C(\{[S] \mid S \in \mathsf{SP}\})$, the same reasoning used above establishes that it is also an element of SHP. Therefore, by mutual containment, the two sets are equal. □

Theorem 5. $(\forall \boldsymbol{P} \in \mathsf{HP} : (\exists \boldsymbol{S} \in \mathsf{SHP}, \boldsymbol{L} \in \mathsf{LHP} : \boldsymbol{P} = \boldsymbol{S} \cap \boldsymbol{L}))$.

Proof. This theorem can be easily proved by adapting either the logical [54] or topological [4] proof of the intersection theorem for trace properties. The domains involved are merely upgraded to include an additional level of sets. Here we take the former approach and rehearse the logical proof.

Our strategy is as follows. Given hyperproperty $\boldsymbol{P}$, we construct safety hyperproperty $\boldsymbol{S}$ that contains $\boldsymbol{P}$ as a subset. We also construct liveness hyperproperty $\boldsymbol{L}$ that contains $\boldsymbol{P}$. The intersection of $\boldsymbol{S}$ and $\boldsymbol{L}$ then necessarily contains $\boldsymbol{P}$, and we shall show that the intersection is, in fact, exactly $\boldsymbol{P}$.

To construct $\boldsymbol{S}$, we define the safety hyperproperty $\mathit{Safe}(\boldsymbol{P})$, which stipulates that the hyperliveness of $\boldsymbol{P}$ is never violated. A bad thing for this safety hyperproperty is any set of traces that cannot be extended to satisfy $\boldsymbol{P}$. So we require that $\mathit{Safe}(\boldsymbol{P})$ contains only sets T of traces such that any observation of T can be extended to satisfy $\boldsymbol{P}$. Formally,

$$\mathit{Safe}(\boldsymbol{P}) \triangleq \{T \in \mathsf{Prop} \mid (\forall M \in \mathsf{Obs} : M \leq T \implies (\exists T' \in \mathsf{Prop} : M \leq T' \wedge T' \in \boldsymbol{P}))\}.$$

It is straightforward to establish that $\mathit{Safe}(\boldsymbol{P})$ is hypersafety: Any set T not contained in $\mathit{Safe}(\boldsymbol{P})$ must satisfy the negation of the predicate in the above definition of $\mathit{Safe}(\boldsymbol{P})$—that is, $(\exists M \in \mathsf{Obs} : M \leq T \ \wedge \ (\forall T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \boldsymbol{P}))$. If no extension of M can be in $\boldsymbol{P}$, then no extension T' of M can be in $\mathit{Safe}(\boldsymbol{P})$ because the hyperliveness of $\boldsymbol{P}$ would be violated in T' at observation M. So

$$\begin{aligned}&(\forall T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \boldsymbol{P})\\ &\qquad\qquad\implies (\forall T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \mathit{Safe}(\boldsymbol{P})).\end{aligned} \tag{D.5}$$

Thus, by monotonicity, $(\exists M \in \mathsf{Obs} : M \leq T \ \wedge \ (\forall T' \in \mathsf{Prop} : M \leq T' \implies T' \notin \mathit{Safe}(\boldsymbol{P})))$. Therefore $\mathit{Safe}(\boldsymbol{P})$ is hypersafety.

Similarly, to construct $\boldsymbol{L}$, we define the liveness hyperproperty $\mathit{Live}(\boldsymbol{P})$, which stipulates that it is always possible either to satisfy $\boldsymbol{P}$ or to become impossible, due to some bad thing, to satisfy $\boldsymbol{P}$. In the latter case, a safety hyperproperty has been violated—namely, $\mathit{Safe}(\boldsymbol{P})$. Formally,

$$\mathit{Live}(\boldsymbol{P}) \ \triangleq\ \boldsymbol{P} \cup \overline{\mathit{Safe}(\boldsymbol{P})},$$

where $\overline{\boldsymbol{H}}$ denotes the complement of hyperproperty $\boldsymbol{H}$ with respect to Prop. To show that $\mathit{Live}(\boldsymbol{P})$ is hyperliveness, consider any observation T. Suppose that T can be extended to some trace property T' such that $T' \in \boldsymbol{P}$. Then T' is also in $\mathit{Live}(\boldsymbol{P})$, so $\mathit{Live}(\boldsymbol{P})$ is hyperliveness for T. On the other hand, if T cannot be extended to satisfy $\boldsymbol{P}$, then T is a bad thing for $\mathit{Safe}(\boldsymbol{P})$—that is, $(\forall T' \in \mathsf{Prop} : T \leq T' \implies T' \notin \boldsymbol{P})$. Let T' be an arbitrary extension of T. By the same reasoning as equation (D.5), T' is not in $\mathit{Safe}(\boldsymbol{P})$. Therefore T' must be in $\overline{\mathit{Safe}(\boldsymbol{P})}$. Thus, $\mathit{Live}(\boldsymbol{P})$ is again hyperliveness for T. We conclude that $\mathit{Live}(\boldsymbol{P})$ is hyperliveness.

Next, note that $\boldsymbol{P} \subseteq \mathit{Safe}(\boldsymbol{P})$, because any element T of $\boldsymbol{P}$ satisfies the definition of $\mathit{Safe}(\boldsymbol{P})$. In particular, for any $M \leq T$, there is a $T' \geq M$ such that $T' \in \boldsymbol{P}$—namely, $T' = T$. Thus, $\mathit{Safe}(\boldsymbol{P}) = \boldsymbol{P} \cup \mathit{Safe}(\boldsymbol{P})$.

Finally, let $\boldsymbol{S} = \mathit{Safe}(\boldsymbol{P})$ and $\boldsymbol{L} = \mathit{Live}(\boldsymbol{P})$, and we prove the theorem by simple set manipulation:

$$\begin{aligned}\boldsymbol{S} \cap \boldsymbol{L} &= \mathit{Safe}(\boldsymbol{P}) \cap \mathit{Live}(\boldsymbol{P})\\ &= (\boldsymbol{P} \cup \mathit{Safe}(\boldsymbol{P})) \cap (\boldsymbol{P} \cup \overline{\mathit{Safe}(\boldsymbol{P})})\\ &= \boldsymbol{P} \cap (\mathit{Safe}(\boldsymbol{P}) \cup \overline{\mathit{Safe}(\boldsymbol{P})})\\ &= \boldsymbol{P} \cap \mathsf{Prop}\\ &= \boldsymbol{P} \qquad\qquad \square\end{aligned}$$