Towards Incrementalization of Holistic Hyperproperties

Dimiter Milushev and Dave Clarke

IBBT-DistriNet, KU Leuven, Heverlee, Belgium

Abstract. A *hyperproperty* is a set of sets of finite or infinite traces over some fixed alphabet and can be seen as a very generic system specification. In this work, we define the notions of *holistic* and *incremental hyperproperties*. Systems specified holistically tend to be more intuitive but difficult to reason about, whereas incremental specifications have a straightforward verification approach. Since most interesting security-related hyperproperties are in the syntactic class of holistic hyperproperties, we introduce the process of *incrementalization* to convert holistic specifications into incremental ones. We then present three incrementalizable classes of holistic hyperproperties and a respective verification method.

1 Introduction

The problem of verifying that a system adheres to some given security policy has been an active research area for several decades. Substantial progress has been made in the verification of security policies that can be expressed as *properties*—first-order predicates over system execution traces. Well-known examples of such progress include the abundance of logics for system specification and the success story of automata-based model checking [7,8,18]. Unfortunately properties are not expressive enough to capture a large class of security policies, such as secure information flow and noninterference.

In an attempt to remedy this and provide a uniform theory of security policies, Clarkson and Schneider formalized security policies as *hyperproperties* [9]. A *hyperproperty* is a second-order predicate over system execution traces, or, in other words, a set of sets of execution traces. Hyperproperties generalize properties and are expressive enough to capture not only secure information flow and noninterference, but also many other interesting policies on systems [9]. Intuitively, a hyperproperty is the set of systems permitted by some policy. (In contrast, a property is the set of runs a system must satisfy.) Although arising in the context of security, hyperproperties are not necessarily limited to security policies; they can be seen as very general and expressive system specifications. Quality of Service (QoS) and Service Level Agreement (SLA) properties can be expressed as hyperproperties.

Clarkson and Schneider specify a class of security-related hyperproperties as first-order predicates on sets of traces [9], using universal and existential quantifiers over traces in a candidate set T, as well as relations on those traces. We call

P. Degano and J.D. Guttman (Eds.): POST 2012, LNCS 7215, pp. 329–348, 2012.

the hyperproperties in this syntactic class *holistic* as they talk about whole traces at once; their specifications tend to be straightforward, but they are difficult to reason about, exemplified by the fact that no general approach to verifying such hyperproperties exists, to the best of our knowledge. To address this problem, we adopt a coalgebraic perspective on systems and hyperproperty specifications and propose to model systems as coalgebras of the functor $GX = 2 \times (1 + X)^A$; this is useful as in their nature, coalgebras and coinductive predicates are "incremental". Intuitively, systems correspond to trees, where the initial state of a system is mapped to the root of the corresponding tree and possible executions build the branches; incremental hyperproperties reason about such trees. Our contributions are firstly formal definitions of the classes of *holistic* and *incremental* hyperproperties, secondly introduction of the notion of *incrementalization* and its application to three classes of holistic hyperproperties, and finally an illustration of the verification approach for incremental hyperproperties.

The rest of the paper is structured as follows. Section 2 provides some background material. Section 3 introduces and formalizes the notions of holistic and incremental hyperproperties and Section 4 presents the incrementalization of three classes of holistic hyperproperties. Section 5 presents a verification method for incremental hyperproperties. Finally, Sections 6 and 7 are left for the related work and conclusion. The proofs can be found in the technical report [21].

2 Background

This section provides the necessary background. Fix a finite alphabet A of abstract observations. A *string* is a finite sequence of elements of A. The set of all strings over A is denoted A^*. A *stream* of A's is an infinite sequence of elements of A. The set of all streams over A is $A^\omega = \{\sigma \mid \sigma : \{0, 1, 2, \ldots\} \to A\}$. A stream σ can be specified in terms of its first element $\sigma(0)$ and its stream derivative σ', given by $\sigma'(n) = \sigma(n + 1)$; these operators are also known as *head* and *tail*. A *trace* is a finite or infinite sequence of elements of A. The set of all traces over A is denoted $A^\infty = A^* \cup A^\omega$. Let 2 be any two element set, for instance the one given as $2 = \{true, false\}$. A *system* is a set of traces. The set of all systems is $\mathsf{Sys} = 2^{A^\infty}$, the set of infinite systems is $\mathsf{Sys}_\omega = 2^{A^\omega}$.

Clarkson and Schneider present a theory of policies based on properties and hyperproperties [9]. A *property* is a set of traces. The set of all properties is $\mathsf{Prop} = 2^{A^\infty}$. A *hyperproperty* is a set of sets of traces or equivalently a set of properties. The set of all hyperproperties is $\mathsf{HP} = 2^{2^{A^\infty}} = 2^{\mathsf{Prop}} = 2^{\mathsf{Sys}}$. Note that our definition, unlike the original one, does not require all traces to be infinite; as a result termination-sensitive definitions can be expressed in a more natural fashion. The *satisfaction relation for hyperproperties* $\models \subseteq \mathsf{Sys} \times 2^{\mathsf{Prop}}$ is defined as $C \models \boldsymbol{H} \;\widehat{=}\; C \in \boldsymbol{H}$. Although $\mathsf{Sys} = \mathsf{Prop}$, we use both names for emphasis.

We now present an example hyperproperty, a variant of *noninterference*. Let $\tau \notin A$ represent unobservable elements of a trace. Let $A_\tau = A \cup \{\tau\}$ and assume predicates *low* and *high* on elements of A such that *low* is equivalent to $\neg$*high*. Coinductively define function $ev_L : A^\infty \to A_\tau^\infty$ to filter out "high" events:

$$\frac{}{ev_L(\epsilon) = \epsilon} \qquad \frac{ev_L(x) = y \qquad high(a)}{ev_L(a \cdot x) = \tau \cdot y} \qquad \frac{ev_L(x) = y \qquad low(a)}{ev_L(a \cdot x) = a \cdot y}$$

Next define *weak trace equivalence* $\approx \subseteq A_\tau^\infty \times A_\tau^\infty$ as:

$$\frac{}{\epsilon \approx \epsilon} \qquad \frac{x \approx y}{\tau \cdot x \approx y} \qquad \frac{x \approx y}{x \approx \tau \cdot y} \qquad \frac{x \approx y}{a \cdot x \approx a \cdot y}$$

Predicate $no_H : A_\tau^\infty \to 2$ states that there are no high events in a trace:

$$\frac{}{no_H(\epsilon)} \qquad \frac{low(a) \qquad no_H(x)}{no_H(a \cdot x)}$$

Finally define *noninterference* as

$$NI = \{T \in \mathsf{Sys} \mid \forall t_0 \in T(\exists t_1 \in T(no_H(t_1) \wedge ev_L(t_0) \approx t_1))\}.$$

For every trace t_0 in a candidate set T the definition of NI requires a low-equivalent modulo weak-bisimulation trace t_1 such that $no_H(t_1)$ is in T. This definition of noninterference is similar in spirit to *strong non-deterministic noninterference* (*NNI*), originally proposed by Focardi and Gorrieri [13]. The major difference is that NI does not distinguish between inputs and outputs; thus it is in a sense stronger than similar definitions that guard the confidentiality of high inputs only (NI ensures the confidentiality of high events, which implies confidentiality of high inputs). Additionally, *NNI* is defined over elements of 2^{A^*}, whereas NI over elements of Sys; finally, *NNI* uses string equality whereas NI uses a form of weak-bisimulation.

Example 1 (Noninterference). Given $A = \{a, b, c\}$, where $high(a)$, $high(c)$, $low(b)$ hold. Consider system $C = \{\sigma, \gamma\}$, where $\sigma = (abc)^\omega$, $\gamma = b^\omega$. Note that $no_H(\gamma) = true$, $ev_L(\sigma) \approx b^\omega$ and $ev_L(\gamma) = b^\omega$ hold. From these we deduce:

1. for σ there exists $t \in C$ s.t. $no_H(t) \wedge ev_L(\sigma) \approx t$, namely $t = \gamma$.
2. for γ there exists $t \in C$ s.t. $no_H(t) \wedge ev_L(\gamma) \approx t$, namely $t = \gamma$.

Hence $C \models NI$: system C satisfies NI as well as variants of *NNI*.

The former definition of noninterference is relatively abstract. In order to additionally illustrate the practical significance of the proposed approach, we also work with *reactive noninterference* [5], a variant of Zdancewic and Myers's definition of *observational determinism* [30] for reactive systems. Without loss of generality, assume that A may be partitioned into A_i and A_o, corresponding to input and output events. Following the original work [5], assume that systems are input-total; moreover, assume that every input event produces some finite or infinite output trace. The model assumes that a system waits for input in some consumer state; whenever an input event is received, the system produces a finite or infinite output trace; if the output trace was finite, the system returns to a consumer state, waiting for further events; otherwise it diverges. For the sake of illustration, we consider deterministic reactive systems. Formally, a *deterministic reactive system RS* can be modeled as the set of traces produced by a function $f_{RS} : A_i^\infty \to A^\infty$. Let $f_i : A_i \to A_o^\infty$ be a function, taking one input event and producing some output trace. Function f_{RS} can be defined coinductively as:

$$\frac{}{f_{RS}(\epsilon) = \epsilon} \qquad \frac{f_i(a) = \sigma \qquad \sigma \in A^* \qquad f_{RS}(r) = \sigma_m}{f_{RS}(a \cdot r) = a \cdot \sigma \cdot \sigma_m} \qquad \frac{f_i(a) = \sigma \qquad \sigma \in A^\omega}{f_{RS}(a \cdot r) = a \cdot \sigma}$$

Let *FRS* be the set of deterministic reactive systems that can be characterized by a functional input-output relation. Let $x \approx_{\mathrm{L}} y$ denote $ev_L(x) \approx ev_L(y)$, and $\approx_{\mathrm{L_i}}$ and $\approx_{\mathrm{L_o}}$ be analogous definitions for input and output events, respectively; similarly $x \approx_{\mathrm{H}} y$ denotes $ev_H(x) \approx ev_H(y)$ and $x \approx_{\mathrm{H_i}} y$ is the restriction to inputs.

Reactive noninterference [5] can be defined as a hyperproperty as follows:

$$RN = \{\, T_f \in FRS \mid \forall t_0, t_1 \in T_f (t_0 \approx_{\mathrm{L_i}} t_1 \rightarrow t_0 \approx_{\mathrm{L}} t_1)\},$$

where $T_f = \{t \in A^\infty \mid \exists \sigma \in A_i^\infty (f(\sigma) = t)\}$. Note that the relation $t_0 \approx_{\mathrm{L}} t_1$ is on whole traces, not on output traces as it is typically defined. This is because $t_0 \approx_{\mathrm{L}} t_1$ implies $t_0 \approx_{\mathrm{L_o}} t_1$, as L_o is a subset of L and the way traces are generated.

Unlike batch-job program models, where all program inputs are available at the start of execution and all program outputs are available at program termination, reactive programs receive inputs and send outputs to their environment during execution. RIMP [5] is a language geared towards writing reactive systems, allowing agents to interact with the system by sending and receiving messages. Messages are typically considered secret to certain agents and public to others. Inputs in RIMP are natural numbers sent over channels and outputs are natural numbers over channels or a tick (τ), signifying an internal action. The channels model users or security levels in some security lattice; typically, L and H model the low and high channel respectively. The detailed syntax and semantics of RIMP are available in the original paper [5]. The following RIMP program illustrates reactive noninterference:

```
1 input ch_H(x) {i := x;}
2 input ch_L(x) {if i <= x then  output ch_L(0);
3                 else output ch_L(1);}
```

Program 1.1. Simple program in RIMP

Let $\sigma_{in} = [ch_H^i(0), ch_L^i(0)]$ and $\gamma_{in} = [ch_H^i(1), ch_L^i(0)]$ be input strings. Clearly $\sigma_{in} \approx_{\mathrm{L_i}} \gamma_{in}$. The traces are $\sigma = [ch_H^i(0), \tau, \tau, ch_L^i(0), \tau, ch_L^o(0), \tau]$ and $\gamma = [ch_H^i(1), \tau, \tau, ch_L^i(0), \tau, ch_L^o(1), \tau]$; because they are not weak trace equivalent at L, it follows that P is not secure, i.e. $P \not\models RN$.

2.1 Partial Automata, Coalgebras and Languages à la Rutten [26]

A *partial automaton* with input alphabet A is defined *coalgebraically* as a 3-tuple $\langle S, o, t\rangle$, where set S is the possibly infinite state space of the automaton, the observation function $o : S \rightarrow 2$ says whether a state is accepting or not, and the partial function $t : S \rightarrow (1 + S)^A$ gives the transition structure. Notation S^A stands for the set of functions with signature $A \rightarrow S$; $1 + S$ is notation used for the set $\{\bot\} \cup S$: whenever the function $t(s)$ is undefined, it is the constant function mapping every undefined symbol from A to $\bot$; if $t(s)$ is defined for some $a \in A$, then $t(s)(a) = s'$ gives the next state. The symbol $\delta \notin A$ is used

to represent *deadlock*. An automaton is in a *deadlock* state s_δ if for all $a \in A$, $t(s_\delta)(a) = \bot$ (the transition function is undefined).

Let $A^* \cdot \delta = \{w \cdot \delta \mid w \in A^*\}$ be the set of finitely deadlocked words. The collection of all languages acceptable by partial automata is $A_\delta^\infty = A^* \cup (A^* \cdot \delta) \cup A^\omega$; note that this can also be seen as a set of sequences i.e. a property, but since any property can be lifted to a hyperproperty [9] there is no inconsistency. Let the truncation of an infinite word $w = a_1 a_2 a_3 \ldots$ to the first n ($n \in \mathbb{N}$) letters be denoted $w[n] = a_1 \ldots a_n$. For words $w \in A^*$ and sets $L \subseteq A_\delta^\infty$, define the *w-derivative* of L to be $L_w = \{v \in A_\delta^\infty \mid w \cdot v \in L\}$. Define a set $L \subseteq A_\delta^\infty$ to be *closed* if for all words w in A^ω, $w \in L \iff \forall n \geq 1, L_{w[n]} \neq \emptyset$. Define a set $L \subseteq A_\delta^\infty$ to be *consistent* if for all words w in A_δ^∞, $\delta \in L_w \iff L_w = \{\delta\}$. The *language of a partial automaton* is a non-empty, closed and consistent subset of A_δ^∞. The set of all such languages is

$$\mathcal{L} = \{L \mid L \subseteq A_\delta^\infty, L \text{ is non-empty, closed and consistent}\}.$$

Any state of a partial automaton accepts some language having three kinds of words: firstly, all finite words that leave the automaton in an accepting state, secondly, all infinite words that cause the automaton to run indefinitely and thirdly, words that lead to a deadlock state. Intuitively, the language of a partial automaton is the language accepted by the start state.

The set $\mathcal{L}$ can be thought of as an automaton $\mathcal{L} = \langle \mathcal{L}, o_\mathcal{L}, t_\mathcal{L} \rangle$ [26]:

$$o_\mathcal{L}(L) = \begin{cases} \mathit{true} & \text{if } \epsilon \in L \\ \mathit{false} & \text{if } \epsilon \notin L \end{cases} \qquad t_\mathcal{L}(L)(a) = \begin{cases} L_a & \text{if } L_a \neq \emptyset \\ \bot & \text{if } L_a = \emptyset. \end{cases}$$

A bisimulation between two automata $S_1 = \langle S, o, t \rangle$ and $S_2 = \langle S', o', t' \rangle$ is a relation $R \subseteq S \times S'$ s.t. for all s in S, s' in S' and a in A

$$s \; R \; s' \implies o(s) = o'(s') \bigwedge t(s)(a) \; (1 + R) \; t'(s')(a).$$

Condition $t(s)(a)$ $(1+R)$ $t'(s')(a)$ holds iff either $t(s)(a) = \bot$ and $t'(s')(a) = \bot$ or $t(s)(a) \; R \; t'(s')(a)$. The maximal bisimulation $\sim$ is the union of all bisimulation relations.

The automaton $\mathcal{L} = \langle \mathcal{L}, o_\mathcal{L}, t_\mathcal{L} \rangle$ satisfies the coinduction proof principle [26]. In other words, for all languages L and K in $\mathcal{L}$ we have: $L \sim K \iff L = K$.

Coalgebras of the polynomial functor $G : Set \to Set$, given by $GX = 2 \times (1 + X)^A$, will be called *G-coalgebras* or *G-systems*. As partial automata are in one-to-one correspondence with G-coalgebras [26], all the theory presented here is applicable to G-coalgebras.

2.2 Systems — From Sets of Traces to *G*-Coalgebras

This section shows that the model of systems as sets of traces can be converted into a tree/coalgebra/partial automaton model; the latter model is more convenient and well-studied. This is an important prerequisite to incrementalization: the conversion of system specifications on sets of traces to specifications on trees.

Let F be an arbitrary functor $F : Set \to Set$. An F-coalgebra is *final* if there is a unique homomorphism from any other F-coalgebra to it. A set of traces can be seen not only as a property or system, but also as a language, and as a G-coalgebra, which itself can be either seen as a partial automaton or as a tree. A *tree* is obtained from a language by continuously taking derivatives with respect to elements of A. Conversely, the language of a tree is given by the paths from the root that either end at a marked node or continue forever. The different perspectives are illustrated in Fig. 1; note that ellipses indicate infinite repetition of the string that has occurred so far. Combined with Rutten's observation [26] that the set of all languages is a final coalgebra, this allows the use of coalgebra and coinduction for reasoning about hyperproperties. We next describe the transition from sets of traces to G-coalgebras.

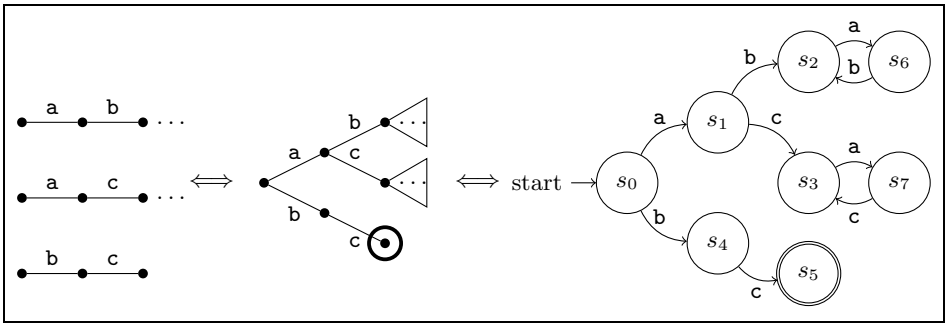

Fig. 1. Equivalent representations of $M = \{(ab)^\omega, (ac)^\omega, bc\}$ over $A = \{a, b, c\}$: traces, trees and G-coalgebras. The branches of the tree are labelled with elements of the alphabet A, its accepting nodes are marked with a circle.

The functor G has the final coalgebra $\mathcal{L}$. The *coinductive definition principle* gives a way to define maps from arbitrary G-coalgebras into the final G-coalgebra. We use the principle to convert an arbitrary set of traces into a G-coalgebra. To this end, take the state space to be $\mathsf{Sys} = 2^{A^\infty}$, the set of all possible sets of traces. Any pair $\langle o, t\rangle$ with signatures $o : \mathsf{Sys} \to 2$ and $t : \mathsf{Sys} \to (1 + \mathsf{Sys})^A$ induces a unique homomorphism $h : \mathsf{Sys} \to \mathcal{L}$ that makes the following diagram commute:

$$\begin{array}{ccc} \mathsf{Sys} & \overset{!h}{\dashrightarrow} & \mathcal{L} \\ \downarrow{\scriptstyle \langle o, t\rangle} & & \downarrow{\scriptstyle \langle o_{\mathcal{L}}, t_{\mathcal{L}}\rangle} \\ 2 \times (1 + \mathsf{Sys})^A & \overset{G(h)}{\dashrightarrow} & 2 \times (1 + \mathcal{L})^A \end{array}$$

Thus the homomorphism h would map any set of traces $s \in \mathsf{Sys}$ to a unique element in the final coalgebra $\mathcal{L}$. Since the elements of the final coalgebra can be seen as trees, we say that h maps a set of traces to the root of a unique tree in $\mathcal{L}$, corresponding precisely to the set of traces.

We now define a particular pair $\langle o, t\rangle$ allowing us to switch perspective from seeing a system as a set of traces to seeing it as a G-coalgebra. Let $C \in \mathsf{Sys}$, $a \in A$ and $\sigma \in A^\infty$. Define an auxiliary function $test : \mathsf{Sys} \to (A \to 2)$ as follows:

$$test_a(C) \;\widehat{=}\; \exists\, \sigma.\sigma \in C \wedge \sigma(0) = a.$$

The functions o and t can be readily defined as follows:

$$o(C) \;\widehat{=}\; \epsilon \in C \qquad t(C)(a) \;\widehat{=}\; \begin{cases} \{\sigma' \mid \sigma(0) = a\} & \text{if } test_a(C) \\ \bot & \text{if } \neg test_a(C). \end{cases}$$

This function pair induces a unique element of the final coalgebra $\mathcal{L}$, corresponding to the inclusion of Sys in $\mathcal{L}$. Clearly every set $S \in \mathsf{Sys}$ is closed and consistent. As a result we may conclude that $\mathsf{Sys} \subseteq \mathcal{L}$.

In summary, any system defined as a set of traces can be uniquely seen as an element (without deadlock states) of the final G-coalgebra, defining its behaviour.

3 Holistic and Incremental Hyperproperties

One of the crucial steps towards verification is finding the class of *incrementalizable* holistic hyperproperties. To that end, we need a formalism for reasoning about holistic and incremental specifications. In this section, we give syntactic definitions of *holistic* and *incremental* hyperproperties. The logical languages used are based on *Least Fixed Point Logic* (*LFP*) [6] — an extension of first order logic by addition of least and greatest fixed point operators. The new logical languages are holistic hyperproperty logic $\mathcal{HL}$, in which most interesting security hyperproperties are expressible, and incremental hyperproperty logic $\mathcal{IL}$. The key difference between the languages is that the former has only coinductive predicates over streams, whereas the latter has only coinductive predicates over systems. In both cases hyperproperties are defined over systems.

3.1 Holistic Hyperproperty Logic $\mathcal{HL}$

Clarkson and Schneider specify hyperproperties as first-order predicates on sets of traces [9], using universal and possibly existential quantification over traces ($\forall t \in T$, $\exists t \in T$) in a candidate set T, as well as relations on tuples of traces. The hyperproperty *NI* (Section 2) is one example. Next, we propose the logical language $\mathcal{HL}$ to formalize hyperproperties specified in this style.

First, we give a grammar for coinductive predicates over streams, a substantial part of $\mathcal{HL}$. Let x range over a set of variables, a over elements of A, p_i over predicates in some set P and X over predicate variables. To define the logic we use standard, trace-manipulation primitives: for any $\sigma \in A^\infty$, $\sigma(0)$ gives the first element (head) of the stream and σ' gives the stream derivative (tail). As usual, $cons : A \times A^\infty \to A^\infty$ is a constructor for streams. The predicates in P have signatures: $p_i : A^{n_i} \to 2$ and $= : A \times A \to 2$. Terms are given as

$$t ::= x \mid \epsilon \mid cons(a, t) \mid t(0) \mid t'$$

and coinductive predicates have the following syntax:

$$\phi_0 ::= p(\overline{t}) \mid X(\overline{t}) \mid \bot \mid \neg\phi_0 \mid \phi_0 \wedge \phi_0,$$

where X can occur only positively in ϕ_0. As usual $\top = \neg\bot$, $\phi \vee \psi = \neg(\neg\phi \wedge \neg\psi)$ and the implication $\phi \rightarrow \psi$ is $\neg\phi \vee \psi$.

Second, define *holistic hyperproperty logic* $\mathcal{HL}$ with the following syntax:

$$\phi_1 ::= \bot \mid \neg\phi_1 \mid \phi_1 \wedge \phi_1 \mid \exists x.\phi_1 \mid x \in X \mid \nu X(\overline{x}).\phi_0.$$

As usual $\forall x.\phi = \neg(\exists x.\neg\phi)$ and $\nu X(\overline{x}).\phi_0$ denotes the greatest fixed point; also note that $x \in X$ iff $X(x) = \mathit{true}$. It is well-known that each coinductive predicate in this language corresponds to a final coalgebra in a category of relations [22]. A *holistic hyperproperty* is a set of sets of traces expressible in $\mathcal{HL}$.

For example, consider defining hyperproperty *FLIP*, assuring that for every stream in a candidate set, its element-wise opposite is also in the set. Start by defining the predicate *flip* $\subseteq A^\omega \times A^\omega$, relating each stream over $A = \{0, 1\}$ to its element-wise opposite:

$$\mathit{flip} \;\widehat{=}\; \nu X(x, y).(\neg(y(0) = x(0)) \wedge X(x', y')).$$

Then, the simple hyperproperty *FLIP* can be given in $\mathcal{HL}$ as:

$$\mathit{FLIP}(X) \;\widehat{=}\; \forall x_0 \in X\ \exists x_1 \in X.\mathit{flip}(x_0, x_1).$$

The proposed logic is fairly general as it captures most security-relevant hyperproperties from the original hyperproperties paper [9]; the noteworthy exceptions are service level agreement (SLA) polices such as mean response time (*MRT*), time service factor and percentage uptime. Formal verification of systems with respect to such policies is an inherently difficult problem [9]: for instance, consider *MRT* as a property; it is generally not clear how to find the mean of a sequence of infinite number of response times in a trace because the series might be diverging; moreover, if *MRT* is seen as a hyperproperty, an additional problem arises, namely that the cardinality of the set of traces might be infinite. We do not address SLA policies in this work.

Next, we present three illustrative examples. First, consider McLean's formulation of a policy called *generalized noninterference* [20] for non-deterministic systems. Informally, the policy states that any high-level behavior is compatible with any low level view of the system. The definition of *GNI* in $\mathcal{HL}$ is

$$\mathit{GNI}(X) \;\widehat{=}\; \forall x_0 \in X\ \forall x_1 \in X\ \exists x_2 \in X.x_2 \approx_{\mathrm{H_i}} x_0 \wedge x_2 \approx_{\mathrm{L}} x_1.$$

Note that $\approx_{\mathrm{H_i}}$ and $\approx_{\mathrm{L}}$ can be defined coinductively and are based on the coinductively defined functions ev_H and ev_L respectively; the latter are not in $\mathcal{HL}$. However, note that combining ev_H and ev_L with $\approx$ gives coinductive predicates.

Second, consider the definition of termination insensitive observational determinism [30]:

$$\mathit{OD}(X) \;\widehat{=}\; \forall x_0 \in X\ \forall x_1 \in X.(x_0(0) =_{\mathrm{L}} x_1(0) \rightarrow x_0 \approx_{\mathrm{L}} x_1),$$

where $=_{\mathrm{L}}$ is an indistinguishability relation on initial program states. Note that the alphabet A is abstract and except events its elements may also be states, as is the case in this example. Third, note that the former definition of OD implies a batch-job model, but for the reactive model of computation, we give the reformulation to RN from Section 2:

$$RN(X) \mathrel{\widehat{=}} \forall x_0 \in X\ \forall x_1 \in X.(x_0 \approx_{\mathrm{L_i}} x_1 \rightarrow x_0 \approx_{\mathrm{L}} x_1).$$

Note that this is one particular definition of termination-insensitive observational determinism. By modifying the definitions of the relation on traces, other flavors of the definitions (e.g. termination-sensitive, time-sensitive) can be obtained.

3.2 Incremental Hyperproperty Logic $\mathcal{IL}$

Incremental hyperproperties can be expressed in a fragment of LFP, called $\mathcal{IL}$ logic. Let y range over a set of tree variables, a, b over alphabet elements and I over predicate variables.

To define the logic, we use the system manipulation primitives $o : \mathsf{Sys} \rightarrow 2$, $(-)_a : \mathsf{Sys} \rightarrow \mathsf{Sys}{+}1$ and $test_a : \mathsf{Sys} \rightarrow 2$ for each $a \in A$: these are the observation, transition and auxiliary $test$ function (see Section 2.2) in the final G-coalgebra. The predicates are $p_i : A^{n_i} \rightarrow 2$, $= : A \times A \rightarrow 2$, o and $test_a$. The terms are

$$T ::= y \mid a \mid T_a.$$

Formulae in $\mathcal{IL}$ have the following syntax:

$$\psi ::= \nu I(\overline{y}).\phi \qquad \phi ::= I(\overline{T}) \mid \bot \mid \neg\phi \mid \phi \wedge \phi \mid \exists a.\phi \mid a \in A \mid p(\bar{T}),$$

where I can occur only positively in ϕ and all occurrences of T_a must be guarded by $test_a(T)$. As an example, consider the coinductive tree predicate $FLIP'(X, Y)$, giving the incremental version of $FLIP$ defined as:

$$FLIP' \mathrel{\widehat{=}} \nu I(X, Y).(\forall a \in A.test_a(X) \rightarrow (\exists b \in A.test_b(Y) \wedge \neg(b = a) \wedge I(X_a, Y_b))).$$

Let Sys^n be the n-ary Cartesian power of Sys. Define an *incremental hyperproperty* to be the greatest fixed point of a monotone function over Sys^n, expressible in $\mathcal{IL}$. In other words, it is a coinductive tree predicate. A hyperproperty $H \in \mathcal{HL}$ is *incrementalizable* iff there exists an $H' \in \mathcal{IL}$ such that for all $T \in \mathsf{Sys}$ we have that $H'(\bar{T}) \equiv H^k(\bar{T})$, where $H^k(\bar{T})$ is an equivalent definition of $H(T)$ on k copies of T. Examples of incrementalizable hyperproperties can be found in Section 4.

The logic is general enough to capture the incremental hyperproperties we are aware of. The long term goal of this work is to *characterize the class of incrementalizable hyperproperties.* As a first step towards this goal, we incrementalize three classes of hyperproperties.

4 Incrementalization of Holistic Hyperproperties

In this section, we outline and illustrate a syntactic approach to incrementalization for three classes of holistic hyperproperties. We give a detailed explanation of the process for the first class.

At a high level, incrementalization of a holistic hyperproperty H, based on trace predicates c_i ($i \in \mathbb{N}$), amounts to finding a coinductive predicate H' and a functional $\Psi_{H'}$ such that H' is the greatest fixed point of $\Psi_{H'}$ (i.e. $H' = \Psi_{H'}(H') = \nu\Psi_{H'}$) and $H(X)$ iff $H'(X, \ldots, X)$ holds. In essence, we lift the fixed point operator, defining coinductive trace predicates in a holistic specification, to the outermost level in an incremental specification. The techniques used include generalizing the definition of H to n parameters, unfolding H and rewriting it using derivatives, unfolding the coinductive definitions c_i, swapping quantifiers, rearranging expressions and folding the holistic definition. This process results in an incremental definition equivalent to H. An incremental hyperproperty, based on a monotone function, corresponds closely to a bisimulation-like notion, from which a verification methodology immediately suggests itself (see Section 5).

4.1 Incrementalization of PHH

Let c be a pointwise, coinductive predicate [22] defined as follows: for $x, y \in A^\infty$ and some functional $R \subseteq A \times A$ (i.e. R can be seen as a function)

$$c \;\widehat{=}\; \nu X(x, y).(x = y = \epsilon) \vee ((x(0)\ R\ y(0)) \wedge X(x', y')).$$

Let PHH be the class of *pointwise, holistic hyperproperties* defined in $\mathcal{HL}$:

$$\mathsf{PHH}(X) \;\widehat{=}\; \forall x \in X\ \exists y \in X.c(x, y).$$

Generalize PHH to take a pair of systems as a parameter as follows:

$$\mathsf{PHH}^2(X, Y) \;\widehat{=}\; \forall x \in X\ \exists y \in Y.c(x, y).$$

Clearly, we have that for all $T \in \mathsf{Sys}$, $\mathsf{PHH}(T)$ iff $\mathsf{PHH}^2(T, T)$. Each of the following lemmas is one or more steps of the incrementalization process. First, unfold the holistic definition of PHH^2.

Lemma 1. *The predicate* $\mathsf{PHH}^2(X, Y)$ *holds iff*

$$\epsilon \in X \rightarrow \epsilon \in Y \bigwedge (\forall a \in A\ \forall w \in A^\infty.\ aw \in X \rightarrow \exists b \in A\ \exists u \in A^\infty.bu \in Y \wedge\ a\ R\ b \wedge c(aw, bu)).$$

Second, rewrite the definition using derivatives and unfold the coinductive definition of c once.

Lemma 2. *The predicate* $\mathsf{PHH}^2(X, Y)$ *holds iff*

$$o(X) \rightarrow o(Y) \bigwedge (\forall a \in A.\ test_a(X) \rightarrow (\forall w \in A^\infty.\ w \in X_a \rightarrow \exists b \in A.\ a\ R\ b \wedge test_b(Y) \wedge \exists u \in A^\infty.\ u \in Y_b \wedge c(w, u))).$$

Third, swap the quantifiers $\exists b$ and $\forall w$; this can be done as b depends only on a.

Lemma 3. *The predicate* $\mathsf{PHH}^2(X, Y)$ *holds iff*

$$o(X) \to o(Y) \bigwedge (\forall a \in A.\ test_a(X) \to (\exists b \in A.\ a\ R\ b \wedge test_b(Y) \wedge (\forall w \in A^\infty.\ w \in X_a \to \exists u \in A^\infty.\ u \in Y_b \wedge c(w, u)))).$$

Fourth, rearrange the resulting expression and fold the definition of PHH^2.

Lemma 4. *The predicate* $\mathsf{PHH}^2(X, Y)$ *holds iff*

$$o(X) \to o(Y) \bigwedge (\forall a \in A.\ test_a(X) \to \exists b \in A.\ test_b(Y) \wedge a\ R\ b\ \wedge \mathsf{PHH}^2(X_a, Y_b)).$$

Finally, define the incremental hyperproperty PIH^2 as follows:

$$\mathsf{PIH}^2 \ \widehat{=}\ \nu I(X, Y).\ (o(X) \to o(Y) \bigwedge (\forall a \in A.\ test_a(X) \to \exists b \in A.\ test_b(Y) \wedge a\ R\ b \wedge I(X_a, Y_b))).$$

Theorem 1 (Incrementalization of PHH^2). *For all* $X, Y \in \mathsf{Sys}$*, we have that* $\mathsf{PHH}^2(X, Y)$ *iff* $\mathsf{PIH}^2(X, Y)$.

Corollary 1. *For all* $T \in \mathsf{Sys}$*, we have that* $\mathsf{PHH}(T)$ iff $\mathsf{PIH}^2(T, T)$.

4.2 Incrementalization of SHH

As illustrated in Section 3.1, a number of security policies can be based on coinductive predicates on traces. We present the incrementalization of SHH — a class of such security-relevant, holistic hyperproperties defined on infinite systems in Sys_ω. This type of definitions claim that a system is secure if the set of traces is closed under removal of high events (see, for instance, [20]). Let $p : A \to 2$ be a predicate and $f : A \to A$ a function. Define a coinductive predicate $\sim_{\mathrm{p}} : A^\omega \times A^\omega \to 2$ as follows:

$$\frac{p(a) \qquad p(b) \qquad x \sim_{\mathrm{p}} y \qquad b = f(a)}{a \cdot x \sim_{\mathrm{p}} b \cdot y} \qquad \frac{\neg p(a) \qquad p(b) \qquad x \sim_{\mathrm{p}} b \cdot y}{a \cdot x \sim_{\mathrm{p}} b \cdot y}$$

$$\frac{p(a) \qquad \neg p(b) \qquad a \cdot x \sim_{\mathrm{p}} y}{a \cdot x \sim_{\mathrm{p}} b \cdot y}$$

Define a coinductive predicate $p_s : A^\omega \to 2$, generalizing no_H from Section 2:

$$\frac{p(a) \qquad p_s(x)}{p_s(a \cdot x)}$$

Let T, X, Y range over Sys_ω. Define SHH as follows:

$$\mathsf{SHH}(X) \ \widehat{=}\ \forall x \in X\ \exists y \in X.\ p_s(y) \wedge x \sim_{\mathrm{p}} y$$

In order to work with only one predicate, combine p_s and $\sim_{\mathrm{p}}$ into a new coinductive predicate c_1:

$$c_1 \;\widehat{=}\; \nu X(x,y).\\ ((p(x(0)) \wedge p(y(0)) \wedge f(x(0)) = y(0) \wedge X(x',y')) \vee (\neg p(x(0)) \wedge X(x',y)))).$$

Lemma 5. *For all $s,t \in A^\omega$, we have that $(p_s(t) \wedge s \sim_{\mathrm{p}} t) \rightarrow c_1(s,t)$.*

To define this class, an additional restriction on Sys_ω is needed. We work only with systems satisfying the property $P_{\Box\Diamond} = \{t \in A^\omega \mid t \models \Box\Diamond p\}$, based on temporal logic modalities *eventually* ($\Diamond$) and *always* ($\Box$) [23].

Lemma 6. *For all $s,t \in P_{\Box\Diamond}$, we have that $c_1(s,t) \rightarrow (p_s(t) \wedge s \sim_{\mathrm{p}} t)$.*

Now, we will work with an equivalent definition of SHH by Lemmas 5 and 6:

$$\mathsf{SHH}(X) \iff \forall x \in X\ \exists y \in X.\ c_1(x,y).$$

Property SHH can be generalized as follows:

$$\mathsf{SHH}^2(X,Y) \;\widehat{=}\; \forall x \in X\ \exists y \in Y.\ c_1(x,y).$$

Clearly, for all $T \in \mathsf{Sys}_\omega$, it is the case that $\mathsf{SHH}(T)$ iff $\mathsf{SHH}^2(T,T)$. Incrementalization allows us to derive the following version of SIH^2:

$$\mathsf{SIH}^2 \;\widehat{=}\; \nu I(X,Y).\ (\forall a \in A.\ test_a(X) \wedge p(a) \rightarrow test_{f(a)}(Y) \wedge I(X_a, Y_{f(a)})\\ \bigwedge \forall a \in A.\ test_a(X) \wedge \neg p(a) \rightarrow I(X_a, Y)).$$

Theorem 2 (Incrementalization of SHH^2). *For all $X,Y \in P_{\Box\Diamond}$, we have that $\mathsf{SHH}^2(X,Y)$ iff $\mathsf{SIH}^2(X,Y)$.*

Corollary 2. *For all $T \in P_{\Box\Diamond}$, we have that $\mathsf{SHH}(T)$ iff $\mathsf{SIH}^2(T,T)$.*

Next, we give some intuition about the restriction for systems to be in the class $P_{\Box\Diamond}$ and argue that it is reasonable. First, note that the predicate $p : A \rightarrow 2$ could be thought of as denoting the visibility of events to agents at a certain security level. Let us call events for which p evaluates to true *p-events*, dually there are *$\neg p$*-events. Intuitively, many security-relevant systems (e.g. reactive systems such as servers) have infinite traces and can be characterized as follows: each trace has some p-event appearing eventually, and that happens infinitely often. These are the type of properties we expect from a server, for instance: each request needs to be eventually serviced and that should happen infinitely often. The latter is captured by the property $P_{\Box\Diamond}$. $P_{\Box\Diamond}$ is a liveness property (informally always possible and possibly infinite), as defined by Alpern and Schneider [3]: formally, such a liveness property is given as follows: $\forall \alpha \in A^*\ \exists \beta \in A^\omega . \alpha\beta \models \Box\Diamond p$.

Thus, the restriction on systems to be in the $P_{\Box\Diamond}$ class is not severe as it actually captures a large class of interesting systems. These are indefinitely running systems in which security is a concern at any point in time.

4.3 Incrementalization of OHH

There are other security policies (such as *timing-sensitive, termination-sensitive noninterference* [2]) based on coinductive predicates on traces. We present the incrementalization of OHH, a class of security-relevant, holistic hyperproperties defined on systems in Sys. Recall that $p : A \to 2$ is a predicate and $f : A \to A$ a function. Define $\sim_{\mathrm{pt}} : A^\infty \times A^\infty \to 2$ on finite or infinite streams:

$$\frac{}{\epsilon \sim_{\mathrm{pt}} \epsilon} \qquad \frac{p(a) \quad p(b) \quad x \sim_{\mathrm{pt}} y \quad b = f(a)}{a \cdot x \sim_{\mathrm{pt}} b \cdot y} \qquad \frac{\neg p(a) \quad \neg p(b) \quad x \sim_{\mathrm{pt}} y}{a \cdot x \sim_{\mathrm{pt}} b \cdot y}$$

Let $\sim_{\mathrm{pt_i}}$ be the natural restriction of the relation on input elements in A_i. Let T, X, Y range over Sys. Define OHH as follows:

$$\mathsf{OHH}(X) \;\widehat{=}\; \forall x \in X\ \forall y \in X.\ (x \sim_{\mathrm{pt_i}} y \to x \sim_{\mathrm{pt}} y).$$

Again, we can generalize OHH to take a pair of systems as a parameter as follows:

$$\mathsf{OHH}^2(X, Y) \;\widehat{=}\; \forall x \in X\ \forall y \in Y.\ (x \sim_{\mathrm{pt_i}} y \to x \sim_{\mathrm{pt}} y).$$

Incrementalization allows to derive the following version of OHH^2:

$$\begin{aligned}\mathsf{OIH}^2 \;\widehat{=}\; & \nu I(X,Y).\ (o(X) \leftrightarrow o(Y)) \\ & \bigwedge \forall a \in A_i.\ test_a(X) \wedge p(a) \wedge test_{f(a)}(Y) \to I(X_a, Y_{f(a)}) \\ & \bigwedge \forall a \in A_o\ \forall b \in A_o.\ test_a(X) \wedge p(a) \wedge test_b(Y) \wedge p(b) \to \\ & \qquad\qquad b = f(a) \wedge I(X_a, Y_{f(a)}) \\ & \bigwedge \forall a \in A\ \forall b \in A.\ test_a(X) \wedge \neg p(a) \wedge test_b(Y) \wedge \neg p(b) \to I(X_a, Y_b)).\end{aligned}$$

Theorem 3 (Incrementalization of OHH^2). *For all $X, Y \in$ Sys, we have that $\mathsf{OHH}^2(X, Y)$ iff $\mathsf{OIH}^2(X, Y)$.*

Corollary 3. *For all $T \in$ Sys, we have that* $\mathsf{OHH}(T)$ iff $\mathsf{OIH}^2(T, T)$.

An interesting, security-related hyperproperty in OHH is weak time-sensitive noninterference [2]. To formalize it, first define its key ingredient $\sim_{\mathrm{ts}}$:

$$\frac{}{\epsilon \sim_{\mathrm{ts}} \epsilon} \qquad \frac{x \sim_{\mathrm{ts}} y \quad low(a)}{a \cdot x \sim_{\mathrm{ts}} a \cdot y} \qquad \frac{x \sim_{\mathrm{ts}} y \quad high(a) \quad high(b)}{a \cdot x \sim_{\mathrm{ts}} b \cdot y}$$

Note that this definition guarantees that both traces terminate in an equal number of steps and are low-view indistinguishable, or both diverge and are still low-view indistinguishable. Thus, the definition is also *termination-sensitive*. Let $\sim_{\mathrm{ts_i}}$ be the same as $\sim_{\mathrm{ts}}$, but restricted to inputs. Finally, define *weak time-sensitive noninterference*:

$$WTSNI(X) \;\widehat{=}\; \forall x \in X\ \forall y \in X.(x \sim_{\mathrm{ts_i}} y \to x \sim_{\mathrm{ts}} y).$$

5 Verification of Incremental Hyperproperties

First, recall some fixed point theory. Let $\Psi : \mathcal{P}(X_1 \times \ldots \times X_n) \to \mathcal{P}(X_1 \times \ldots \times X_n)$ be a monotone operator over the complete lattice of set-theoretic n-ary relations on the Cartesian product of sets X_i, $i \in 1..n$, and $gfp(\Psi)$ be the greatest fixed point of Ψ. For any $\overline{x} = (x_1, \ldots, x_n) \in X_1 \times \ldots \times X_n$ and $R \subseteq X_1 \times \ldots \times X_n$ the following principle is sound:

$$\frac{R(\overline{x}) \qquad R \subseteq \Psi(R)}{gfp(\Psi)(\overline{x})}$$

This is the *Tarski coinduction principle* [17], generalised to an n-ary relation.

Due to the nature of incrementalization, the original holistic definition H becomes irrelevant for verification. Instead, the resulting coinductive predicate H' has to be verified to show that H holds. One way to do this is via theory established by Niqui and Rutten [22]. For a coinductive predicate H', they introduce *H'-simulations* which are to coinductive predicates as bisimulations are to equality. Formally an H'-simulation is an n-ary relation R such that $R \subseteq \Psi_{H'}(R)$: an H'-simulation corresponds closely to a respective monotone operator $\Psi_{H'}$ (see Section 3.2) whose greatest fixed point is H'. Thus, finding an H'-simulation for the system of interest will be sufficient for showing that H' holds.

Verification of an incremental hyperproperty would typically have two steps. First, find an appropriate notion of H'-simulation. Second, find a specific H'-simulation for the system of interest. Showing that there is no H'-simulation implies that the predicate does not hold. The second step can be automatic, adapting techniques from automata-based model checking. The soundness of this verification methodology follows directly from Tarski's coinduction principle [17].

Theorem 4. *The predicate $H'(x_1, \ldots, x_k)$ on G-systems $\langle S_i, \alpha_i, x_i \rangle$ for $i \in 1..k$ holds iff there exists some H'-simulation Q s.t. the k-tuple of the start states $\langle x_1, \ldots, x_k \rangle \in Q$.*

It should be noted that Theorem 4 is not constructive: it does not give an algorithm for finding H'-simulations. Nevertheless, automata-based model checking techniques provide a practical means to compute or approximate the greatest fixed point of a monotone operator. There are well-known iterative schemata for computing the greatest fixed points on a complete lattice [29]. Typically, the bottom and top elements of the lattice are the empty set and the powerset of the state space S, the partial order relation is set inclusion. When approximating (or calculating for finite state spaces) the greatest fixed point one starts with S and iteratively applies the functional Ψ. Formally $\Psi^0 = id$ and $\Psi^{n+1} = \Psi \circ \Psi^n$ where id is the identity function and operator $\circ$ denotes composition. The greatest fixed point of Ψ can be found as follows: $\nu\Psi = \bigcap_{n \geq 0} \Psi^n(S)$. The technique needs adaptation in order to accommodate our different state space, namely $S_1 \times \ldots \times S_k$. Finding efficient algorithms for deciding whether concrete coinductive predicates hold has not been explored yet and is left for future work. Nevertheless, the fact that different notions of bisimilarity can be decided more efficiently than language equivalence on finite transition systems [1] is promising.

5.1 Sample Hyperproperties in PHH

Consider the holistic hyperproperty *FLIP* from Section 3.1, which is in the class PHH. Therefore, we can instantiate the incremental definition of PIH^2 and find the appropriate notion of $FLIP'$-simulation: a relation Q is a $FLIP'$-simulation whenever, if $y_1\ Q\ y_2$, then

$$o(y_1) \to o(y_2)$$
$$\bigwedge \forall a \in A.\ test_a(y_1) \to \exists b \in A.\ test_b(y_2) \wedge b = \neg a \wedge t(y_1)(a)\ Q\ t(y_2)(b).$$

Note that we rely on the fact that there is a unique map from G-coalgebras to trees. Now, take the system ζ_1 represented by the automaton in Fig. 2a. Relation $Q = \{(s_0, s_0), (s_0, s_1), (s_1, s_0)\}$ is the needed $FLIP'$-simulation and thus $\zeta_1 \models FLIP$.

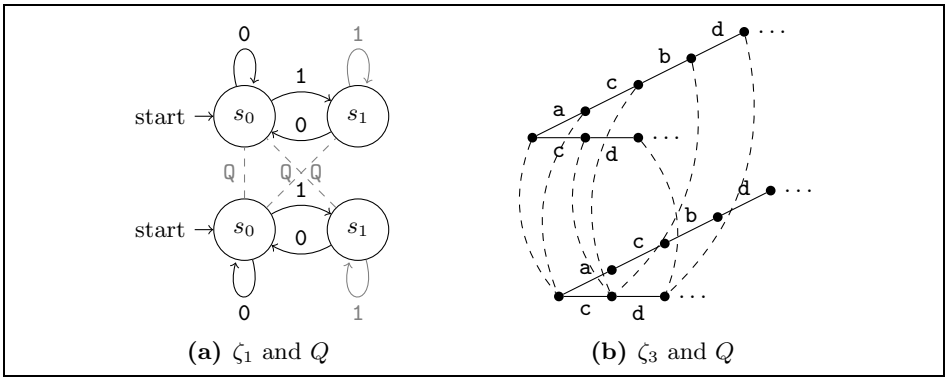

Fig. 2. Illustration of $FLIP'$-simulation in (a) and NI'-simulation in (b)

Let ζ_2 be the system resulting from the removal of the transition from state s_1 to itself (gray transition in Fig. 2a). We show that there is no $FLIP'$-simulation Q such that $\langle s_0, s_0\rangle \in Q$. To that end, assume there were such Q. By the assumption and the definition of $FLIP'$-simulation we have that the pair $\langle t(s_1)(1), t(s_0)(0)\rangle$ should be in Q (and in every $FLIP'$-simulation). This is not the case as $t(s_1)(1) = \bot$, whereas $t(s_0)(0) = s_0$. This is a contradiction and thus there is no Q that is a $FLIP'$-simulation and $\langle s_0, s_0\rangle \in Q$. Thus $\zeta_2 \not\models FLIP$.

5.2 Sample Hyperproperties in SHH

Consider *NI* from Section 2 on systems satisfying $P_{\Box\Diamond}$, which is in the class SHH. Hence we can instantiate the definition of SIH^2 and get an incremental definition of *NI*, called as usual NI' and corresponding to an NI'-simulation: a relation Q such that if $y_1\ Q\ y_2$, then

$$\forall a \in A.\ (test_a(X) \wedge low(a) \to test_a(Y) \wedge t(y_1)(a)\ Q\ t(y_2)(a))$$
$$\bigwedge \forall a \in A.\ (test_a(X) \wedge high(a) \to t(y_1)(a)\ Q\ y_2).$$

Next, consider alphabet $A = \{a, b, c, d\}$ and define predicate *high* as $high(a)$, $high(b)$, $\neg high(c)$, $\neg high(d)$. Take the system $\zeta_3 = \{(acbd)^\omega, (cd)^\omega\}$ (see Fig. 2b). Let $Q = \{(T, T), (T_a, T), (T_{ac}, T_c), (T_{acb}, T_c),\ (T_{acbd}, T_{cd}), (T_c, T_c), (T_{cd}, T_{cd})\}$. Q is an NI'-simulation such that $\langle T, T\rangle \in Q$ and thus we conclude that $\zeta_3 \models NI$.

5.3 Sample Hyperproperties in OHH

Recall hyperproperty *WTSNI* from the class OHH. We can get an incremental definition, corresponding to a $WTSNI'$-simulation: a relation Q s.t. if $y_1\ Q\ y_2$

$$
\begin{aligned}
& o(y_1) \leftrightarrow o(y_2) \\
& \bigwedge \forall a \in A_i.\ test_a(X) \wedge low(a) \wedge test_a(Y) \rightarrow t(y_1)(a)\ Q\ t(y_2)(a) \\
& \bigwedge \forall a \in A_o\ \forall b \in A_o.\ test_a(X) \wedge low(a) \wedge test_b(Y) \wedge low(b) \rightarrow \\
& \qquad\qquad a = b \wedge t(y_1)(a)\ Q\ t(y_2)(a) \\
& \bigwedge \forall a \in A\ \forall b \in A. test_a(X) \wedge high(a) \wedge test_b(Y) \wedge high(b) \rightarrow \\
& \qquad\qquad t(y_1)(a)\ Q\ t(y_2)(b).
\end{aligned}
$$

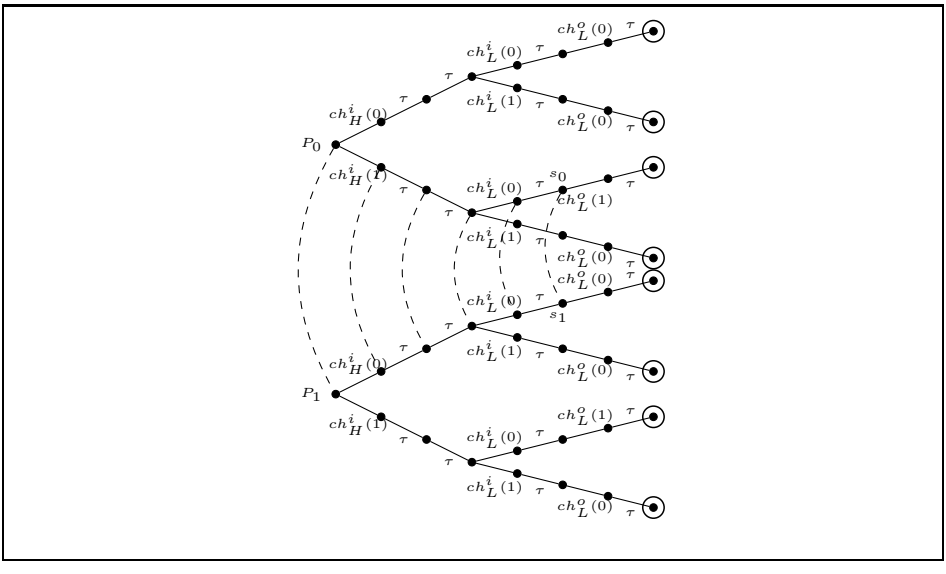

Fig. 3. Illustration of the lack of $WTSNI'$-simulation for Program 1.1

To illustrate the applicability of our abstract notions to programs, consider the RIMP Program 1.1, also called P. Two copies of the program (P_0 and P_1) are presented visually in Fig. 3. We show that there is no $WTSNI'$-simulation

Q such that $\langle P_0, P_1\rangle \in Q$; to that end assume there were such a Q; states that should be related in any $WTSNI'$-simulation are connected by the dashed line. By the assumption and the definition of $WTSNI'$-simulation, it follows that $\langle t(P_0)(ch^i_H(1)), t(P_1)(ch^i_H(0))\rangle \in Q$. This process continues until the pair $\langle s_0, s_1\rangle$ is reached. The definition requires that $ch^o_L(1) = ch^o_L(0)$, by the rule for low outputs. This is a contradiction, thus there is no Q that is a $WTSNI'$-simulation and $\langle P_0, P_1\rangle \in Q$. Hence $P \not\models WTSNI$. Note that Program 1.1 is also insecure with respect to other, less strict definitions, such as RN (see Section 2).

To illustrate that this definition is termination-sensitive, consider Program 1.2, also called P_2, which is termination-insensitive noninterferent. To see this, let $\sigma_{in} = [ch^i_H(1)]$ and $\gamma_{in} = [ch^i_H(0)]$ be input strings. Clearly, $\sigma_{in} \approx_{\mathrm{L}} \gamma_{in}$ and the resultant traces are $\sigma = [ch^i_H(1), \tau, \tau, \tau, ch^o_L(0), \tau]$ and $\gamma = [ch^i_H(0), \tau, \tau, \tau, \ldots]$. Since $\sigma \approx_{\mathrm{L}} \gamma$, and these are all traces, it follows that P_2 is secure. The application of the theory to this example is relatively straightforward and left to the reader.

```
1 input ch_H(x) {i:=x; if i = 0 then while 1 do skip};
2 output ch_L(0);
```

Program 1.2. Termination-sensitive interferent program in RIMP

6 Related Work

Clarkson and Schneider show that labelled transition systems can be encoded as sets of traces [10]. They argue that bisimulation-based hyperproperties, notably Focardi and Gorrieri's *bisimulation nondeducibility on composition* ($BNDC$), can be converted into trace sets. This is in effect the opposite to what we suggest. We propose going from trace sets to state-based systems because the latter are well-understood and enjoy well-established verification techniques, as well as mature verification tools. It is arguable, but we believe that such an approach is more natural and generic enough to work for a large number of applications.

That incrementalization is useful can be seen from recent work on noninterference for reactive systems [5]; Bohannon et al. start with a holistic definition of reactive noninterference and convert it into a relation on program states that they call *ID-bisimulation*; they effectively make the definition incremental. The authors use that latter incremental definition in order to prove that well-typed RIMP programs are secure. They also show that an *ID-bisimulation* implies the high level, holistic policy.

At first sight, incrementalization is somewhat similar to *unwinding* [14]. As Goguen and Meseguer describe it, unwinding is the process of translating a security policy first into local constraints on the transition system that inductively guarantee that the policy is satisfied and second in a finite set of lemmas; any system that satisfies the lemmas is guaranteed to satisfy the policy. The main difference to our work is that unwinding is still a trace based property, whereas incrementalization results in coinductive predicates and reasoning on trees. In addition, incrementalization gives an equivalent definition of the hyperproperty, whereas unwinding gives only a logically sufficient condition.

It turns out that incremental hyperproperties are inherently related to Mantel's work on unwinding of possibilistic security properties [19]. He proposes a modular framework in which most well-known security properties can be composed from a set of *basic security properties* (BSPs); he also presents unwinding conditions for most BSPs. His unwinding conditions are specified locally on states of the system (inspired by Rushby's work [25]) as opposed to the more traditional global (trace-based) unwinding conditions. These unwinding conditions can be seen as simulation relations on system states and in that sense are similar to our incremental security hyperproperties. A major difference is that Mantel's traces are only finite, i.e. his systems are in 2^{A^*}. Mantel also shows that the unwinding conditions for his BSPs are generally sound and only complete for a restricted class of models, in which any event is either high or low. In summary, Mantel's unwinding conditions can be seen as instances of incremental hyperproperties on finite systems.

Recent work [12] has proposed an automata-theoretic technique for model checking the possibilistic information flow hyperproperties from Mantel's framework [19] on finite state systems. To that end the authors show how to model check Mantel's BSPs, which are the building blocks of the respective holistic hyperproperties. This is a nice theoretical result, supporting our thesis that incremental hyperproperties are amenable to model checking. On the negative side, the authors show that the model checking problem is undecidable for the class of pushdown systems. Although using unwinding conditions (simulation relations), the proposed model checking approach is based on deciding set inclusion on regular languages. The latter question can be answered by standard automata-theoretic techniques. Such an approach is not directly applicable to hyperproperties, because the presence of infinite traces means that the languages (sets of traces) under consideration are not regular.

Also in recent work, Huisman and Blondeel [16] give a modal μ-calculus characterization of two determinism-based notions of information flow: observational determinism and eager trace equivalence [24]. Their characterization is based on a self-composed model [4,11] of the transition system induced by the program of interest; effectively the program would be executed in parallel with itself. The major differences to our framework is that we are not restricted to deterministic systems, thus we can handle more security-relevant hyperproperties.

There has been a substantial amount of work on verifying other specific hyperproperties, most notably of secure information flow from both the language-based security [28] and process calculi security [13,27] communities. Language-based secure information flow has traditionally relied on information flow type systems, with a recent trend to incorporate program logics or a combination of both [28,4,11,15]. There have also been attempts to address noninterference using results from process algebra [27,13]. Common for this line of work is formalizing different definitions of security and showing that they all depend on some notion of equivalence of processes, e.g. strong, weak, power bisimulation.

7 Conclusion

This work presents a formal classification of hyperproperties into holistic and incremental ones. It furthermore motivates the shift from a holistic to an incremental approach to hyperproperty specifications, dictated by the fact that the incremental approach has a clearer verification methodology. We argue that identifying the class of hyperproperties that are incrementalizable and finding a generic methodology for incrementalization of holistic hyperproperties are important problems. We propose a generic framework and techniques to explore the process of incrementalization and the usefulness of the resulting incremental hyperproperties. We identify three classes of incrementalizable hyperproperties. Future work will explore the problems presented here as well as techniques for model checking incremental hyperproperties.

Acknowledgements. We would like to thank Frank Piessens and José Proenca for valuable comments on drafts of this paper. We also thank the anonymous reviewers for their constructive feedback.

References

1. Aceto, L., Ingolfsdottir, A., Srba, J.: The Algorithmics of Bisimilarity. In: Advanced Topics in Bisimulation and Coinduction, pp. 100–172. Cambridge University Press (2011)
2. Agat, J.: Transforming out timing leaks. In: Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2000, pp. 40–53. ACM, New York (2000)
3. Alpern, B., Schneider, F.B.: Defining liveness. Technical report, Ithaca, NY, USA (1984)
4. Barthe, G., D'Argenio, P.R., Rezk, T.: Secure information flow by self-composition. In: CSFW 2004: Proceedings of the 17th IEEE Workshop on Computer Security Foundations, p. 100. IEEE Computer Society, Washington, DC (2004)
5. Bohannon, A., Pierce, B.C., Sjöberg, V., Weirich, S., Zdancewic, S.: Reactive noninterference. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 79–90. ACM, New York (2009)
6. Bradfield, J., Stirling, C.: Modal mu-calculi. In: Handbook of Modal Logic, pp. 721–756. Elsevier (2007)
7. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems 8(2), 244–263 (1986)
8. Clarke Jr., E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge (1999)
9. Clarkson, M.R., Schneider, F.B.: Hyperproperties. In: CSF 2008: Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium, pp. 51–65. IEEE Computer Society, Washington, DC (2008)
10. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security 18, 1157–1210 (2010)
11. Darvas, Á., Hähnle, R., Sands, D.: A Theorem Proving Approach to Analysis of Secure Information Flow. In: Hutter, D., Ullmann, M. (eds.) SPC 2005. LNCS, vol. 3450, pp. 193–209. Springer, Heidelberg (2005)

12. D'Souza, D., Holla, R., Raghavendra, K.R., Sprick, B.: Model-checking trace-based information flow properties. Journal of Computer Security 19, 101–138 (2011)
13. Focardi, R., Gorrieri, R.: A taxonomy of security properties for process algebras. Journal of Computer Security 3(1), 5–34 (1995)
14. Goguen, J.A., Meseguer, J.: Unwinding and inference control. In: IEEE Symposium on Security and Privacy, p. 75 (1984)
15. Hähnle, R., Pan, J., Rümmer, P., Walter, D.: Integration of a Security Type System into a Program Logic. In: Montanari, U., Sannella, D., Bruni, R. (eds.) TGC 2006. LNCS, vol. 4661, pp. 116–131. Springer, Heidelberg (2007)
16. Huisman, M., Blondeel, H.-C.: Model-Checking Secure Information Flow for Multi-threaded Programs. In: Mödersheim, S., Palamidessi, C. (eds.) TOSCA 2011. LNCS, vol. 6993, pp. 148–165. Springer, Heidelberg (2012)
17. Lenisa, M.: From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. Electronic Notes in Theoretical Computer Science, 19 (1999)
18. Manna, Z., Pnueli, A.: The temporal logic of reactive and concurrent systems. Springer-Verlag New York, Inc., New York (1992)
19. Mantel, H.: Unwinding Possibilistic Security Properties. In: Cuppens, F., Deswarte, Y., Gollmann, D., Waidner, M. (eds.) ESORICS 2000. LNCS, vol. 1895, pp. 238–254. Springer, Heidelberg (2000)
20. McLean, J.: A general theory of composition for a class of possibilistic properties. IEEE Transactions on Software Engineering 22(1), 53–67 (1996)
21. Milushev, D., Clarke, D.: Towards incrementalization of holistic hyperproperties: extended version. Technical Report CW 616, Katholieke Universiteit Leuven (December 2011)
22. Niqui, M., Rutten, J.: Coinductive predicates as final coalgebras. In: Matthes, R., Uustalu, T. (eds.) Proceedings of the 6th Workshop on Fixed Points in Computer Science, FICS 2009, Coimbra, Portugal, September 12-13, pp. 79–85 (2009)
23. Pnueli, A.: The temporal semantics of concurrent programs. In: Proceedings of the International Sympoisum on Semantics of Concurrent Computation, pp. 1–20. Springer, London (1979)
24. Roscoe, A.W.: CSP and determinism in security modelling. In: Proceedings of the 1995 IEEE Symposium on Security and Privacy, SP 1995, pp. 114–127. IEEE Computer Society, Washington, DC (1995)
25. Rushby, J.: Noninterference, transitivity and channel-control security policies. Technical report (1992)
26. Rutten, J.J.M.M.: Automata and Coinduction (an Exercise in Coalgebra). In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 194–218. Springer, Heidelberg (1998)
27. Ryan, P.Y.A., Schneider, S.A.: Process algebra and non-interference. Journal of Computer Security 9(1/2), 75–103 (2001)
28. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications 21(1), 5–19 (2003)
29. Stirling, C.: Modal and temporal properties of processes. Springer-Verlag New York, Inc., New York (2001)
30. Zdancewic, S., Myers, A.C.: Observational determinism for concurrent program security. In: IEEE Computer Security Foundations Workshop, p. 29 (2003)