Quantitative Information Flow as Safety and Liveness Hyperproperties*

Hirotoshi Yasuoka
Tohoku University
Sendai, Japan
yasuoka@kb.ecei.tohoku.ac.jp

Tachio Terauchi
Nagoya University
Nagoya, Japan
terauchi@is.nagoya-u.ac.jp

We employ Clarkson and Schneider's "hyperproperties" to classify various verification problems of quantitative information flow. The results of this paper unify and extend the previous results on the hardness of checking and inferring quantitative information flow. In particular, we identify a subclass of liveness hyperproperties, which we call "k-observable hyperproperties", that can be checked relative to a reachability oracle via self composition.

1 Introduction

We consider programs containing high security inputs and low security outputs. Informally, the quantitative information flow problem concerns the amount of information that an attacker can learn about the high security input by executing the program and observing the low security output. The problem is motivated by applications in information security. We refer to the classic by Denning [13] for an overview.

In essence, quantitative information flow measures *how* secure, or insecure, a program (or a part of a program –e.g., a variable–) is. Thus, unlike non-interference [11, 15], that only tells whether a program is completely secure or not completely secure, a definition of quantitative information flow must be able to distinguish two programs that are both interfering but have different levels of security.

For example, consider the programs $M_1 \equiv \texttt{if}\ H = g\ \texttt{then}\ O := 0\ \texttt{else}\ O := 1$ and $M_2 \equiv O := H$. In both programs, H is a high security input and O is a low security output. Viewing H as a password, M_1 is a prototypical login program that checks if the guess g matches the password. By executing M_1, an attacker only learns whether H is equal to g, whereas she would be able to learn the entire content of H by executing M_2. Hence, a reasonable definition of quantitative information flow should assign a higher quantity to M_2 than to M_1, whereas non-interference would merely say that M_1 and M_2 are both interfering, assuming that there are more than one possible value of H.

Researchers have attempted to formalize the definition of quantitative information flow by appealing to information theory. This has resulted in definitions based on the Shannon entropy [13, 8, 21], the min entropy [28], and the guessing entropy [17, 3]. All of these definitions map a program (or a part of a program) onto a non-negative real number, that is, they define a function $\mathscr{X}$ such that given a program M, $\mathscr{X}(M)$ is a non-negative real number. (Concretely, $\mathscr{X}$ is $SE[\mu]$ for the Shannon-entropy-based definition with the distribution μ, $ME[\mu]$ for the min-entropy-based definition with the distribution μ, and $GE[\mu]$ for the guessing-entropy-based definition with the distribution μ.)

In a previous work [32, 31], we have proved a number of hardness results on checking and inferring quantitative information flow (QIF) according to these definitions. A key concept used to connect the hardness results to QIF verification problems was the notion of k-safety, which is an instance in a

*This work was supported by MEXT KAKENHI 23700026, 22300005, 23220001, and Global COE Program "CERIES."

To appear in EPTCS.

	$SE[U]$	$ME[U]$	$GE[U]$
LBP	Liveness	Liveness	Liveness
UBP	Liveness	Safety	Safety
LBP constant bound	Liveness	k-observable	k-observable
UBP constant bound	Liveness	k-safety [31]	k-safety [31]

Table 1: A summary of hyperproperty classifications

collection of the class of program properties called *hyperproperties* [10]. In this paper, we make the connection explicit by providing a fine-grained classification of QIF problems, utilizing the full range of hyperproperties. This has a number of benefits, summarized below.

1.) A unified view on the hardness results of QIF problems.

2.) New insights into hyperproperties themselves.

3.) A straightforward derivation of some complexity theoretic results.

Regarding 1.), we focus on two types of QIF problems, an upper-bounding problem that checks if QIF of a program is bounded above by the given number, and a lower-bounding problem that checks if QIF is bounded below by the given number. Then, for each QIF definitions *SE*, *GE*, *ME*, we classify whether or not they are safety hyperproperty, k-safety hyperproperty, liveness hyperproperty, or k-observable hyperproperty (and give a bound on k for k-safe/k-observable). Safety hyperproperty, k-safety hyperproperty, liveness hyperproperty, and observable hyperproperty are classes of hyperproperties defined by Clarkson and Schneider [10]. In this paper, we identify new classes of hyperproperties, k-observable hyperproperty, that is useful for classifying QIF problems. k-observable hyperproperty is a subclass of observable hyperproperties, and observable hyperproperty is a subclass of liveness hyperproperties.[1] We focus on the case the input distribution is uniform, that is, $\mu = U$, as showing the hardness for a specific case amounts to showing the hardness for the general case. Also, checking and inferring QIF under the uniformly distributed inputs has received much attention [16, 3, 18, 7, 21, 8], and so, the hardness for the uniform case is itself of research interest.[2]

Regarding 2.), we show that the k-observable subset of the observable hyperproperties is amenable to verification via self composition [4, 12, 29, 25, 30], much like k-safety hyperproperties, and identify which QIF problems belong to that family. We also show that the hardest of the QIF problems (but nevertheless one of the most popular) can only be classified as a general liveness hyperproperty, suggesting that liveness hyperproperty is a quite permissive class of hyperproperties.

Regarding 3.), we show that many complexity theoretic results for QIF problems of loop-free boolean programs can be derived from their hyperproperties classifications [32, 31]. We also prove new complexity theoretic results, including the (implicit state) complexity results for loop-ful boolean programs, complementing the recently proved explicit state complexity results [6].

Table 1 and Table 2 summarize the hyperproperties classifications and computational complexities of upper/lower-bounding problems. We abbreviate lower-bounding problem, upper-bounding problem, and boolean programs to LBP, UBP, and BP, respectively. The "constant bound" rows correspond to bounding problems with a constant bound (whereas the plain bounding problems take the bound as an input).

The proofs omitted from the main body of the paper appear in the Appendix.

[1] Technically, only non-empty observable hyperproperties are liveness hyperproperties.

[2] In fact, computing QIF under other input distributions can sometimes be reduced to this case [2]. See also Section 5.3.

	$SE[U]$	$ME[U]$	$GE[U]$
LBP for BP	*PSPACE*-hard	*PSPACE*-complete	*PSPACE*-complete
UBP for BP	*PSPACE*-hard	*PSPACE*-complete	*PSPACE*-complete
LBP for loop-free BP	*PP*-hard	*PP*-hard	*PP*-hard
UBP for loop-free BP	*PP*-hard [31]	*PP*-hard [31]	*PP*-hard [31]
LBP for loop-free BP, constant bound	Unknown	*NP*-complete	*NP*-complete
UBP for loop-free BP, constant bound	Unknown	*coNP*-complete	*coNP*-complete

Table 2: A summary of computational complexities

2 Preliminaries

2.1 Quantitative Information Flow

We introduce the information theoretic definitions of QIF that have been proposed in literature. First, we review the notion of the *Shannon entropy* [27], $\mathscr{H}[\mu](X)$, which is the average of the information content, and intuitively, denotes the uncertainty of the random variable X. And, we review the notion of the *conditional entropy*, $\mathscr{H}[\mu](Y|Z)$, which denotes the uncertainty of Y after knowing Z.

Definition 2.1 (Shannon Entropy and Conditional Entropy) *Let X be a random variable with sample space $\mathbb{X}$ and μ be a probability distribution associated with X (we write μ explicitly for clarity). The Shannon entropy of X is defined as*

$$\mathscr{H}[\mu](X) = \sum_{x \in \mathbb{X}} \mu(X = x) \log \frac{1}{\mu(X = x)}$$

Let Y and Z be random variables with sample space Y and Z, respectively, and μ' be a probability distribution associated with Y and Z. Then, the conditional entropy of Y given Z is defined as

$$\mathscr{H}[\mu](Y|Z) = \sum_{z \in \mathbb{Z}} \mu(Z = z) \mathscr{H}[\mu](Y|Z = z)$$

where

$$\mathscr{H}[\mu](Y|Z = z) = \textstyle\sum_{y \in \mathbb{Y}} \mu(Y = y|Z = Z) \log \frac{1}{\mu(Y=y|Z=z)}$$
$$\mu(Y = y|Z = z) = \frac{\mu(Y=y,Z=z)}{\mu(Z=z)}$$

(The logarithm is in base 2.)

Let M be a program that takes a high security input H, and gives the low security output trace O. For simplicity, we restrict to programs with just one variable of each kind, but it is trivial to extend the formalism to multiple variables (e.g., by letting the variables range over tuples or lists). Also, for the purpose of the paper, unobservable (i.e., high security) output traces are irrelevant, and so we assume that the only program output is the low security output trace. Let μ be a probability distribution over the values of H. Then, the semantics of M can be defined by the following probability equation. (We restrict to deterministic programs in this paper.)

$$\mu(O = o) = \sum_{\substack{h \in \mathbb{H} \\ M(h) = o}} \mu(H = h)$$

Here, $M(h)$ denotes the infinite low security output trace of the program M given a input h, and $M(h) = o$ denotes the output trace of M given h that is equivalent to o. In this paper, we adopt the termination-insensitive security observation model, and let the outputs o and o' be equivalent iff $\forall i \in \omega.o_i = \bot \vee o'_i = \bot \vee o_i = o'_i$ where o and o_i denotes the ith element of o, and $\bot$ is the special symbol denoting termination.[3]

In this paper, programs are represented by sets of traces, and traces are represented by lists of stores of programs. More formally,

$$M(h) \text{ is equal to } o \quad \text{iff} \quad \sigma_0;\sigma_1;\ldots;\sigma_i;\ldots \in M$$
$$\text{where } \sigma_0(H) = h \text{ and } \forall i \geq 1.\sigma_i(O) = o_i \ (o_i \text{ denotes the ith element of } o)$$

Here, σ denotes a store that maps variables to values. Because we restrict all programs to deterministic programs, every program M satisfies the following condition: For any trace $\overrightarrow{\sigma}, \overrightarrow{\sigma}' \in M$, we have $\sigma_0(H) = \sigma'_0(H) \Rightarrow \overrightarrow{\sigma} = \overrightarrow{\sigma}'$ where σ_0 and σ'_0 denote the first elements of $\overrightarrow{\sigma}$ and $\overrightarrow{\sigma}'$, respectively. Now, we are ready to define Shannon-entropy-based quantitative information flow.

Definition 2.2 (Shannon-Entropy-based QIF [13, 8, 21]) *Let M be a program with a high security input H, and a low security output trace O. Let μ be a distribution over H. Then, the Shannon-entropy-based quantitative information flow is defined*

$$SE[\mu](M) = \mathscr{H}[\mu](H) - \mathscr{H}[\mu](H|O)$$

Intuitively, $\mathscr{H}[\mu](H)$ denotes the initial uncertainty and $\mathscr{H}[\mu](H|O)$ denotes the remaining uncertainty after knowing the low security output trace. (For space, the paper focuses on the low-security-input free definitions of QIF.)

As an example, consider the programs M_1 and M_2 from Section 1. For concreteness, assume that g is the value 01 and H ranges over the space $\{00,01,10,11\}$. Let U be the uniform distribution over $\{00,01,10,11\}$, that is, $U(h) = 1/4$ for all $h \in \{00,01,10,11\}$. The results are as follows.

$$SE[U](M_1) = \mathscr{H}[U](H) - \mathscr{H}[U](H|O) = \log 4 - \tfrac{3}{4}\log 3 \approx .81128$$

$$SE[U](M_2) = \mathscr{H}[U](H) - \mathscr{H}[U](H|O) = \log 4 - \log 1 = 2$$

Consequently, we have that $SE[U](M_1) \leq SE[U](M_2)$, but $SE[U](M_2) \not\leq SE[U](M_1)$. That is, M_1 is more secure than M_2 (according to the Shannon-entropy based definition with uniformly distributed inputs), which agrees with our intuition.

Next, we introduce the *min entropy*, which has recently been suggested as an alternative measure for quantitative information flow [28].

Definition 2.3 (Min Entropy) *Let X and Y be random variables, and μ be an associated probability distribution. Then, the min entropy of X is defined*

$$\mathscr{H}_\infty[\mu](X) = \log \frac{1}{\mathscr{V}[\mu](X)}$$

and the conditional min entropy of X given Y is defined

$$\mathscr{H}_\infty[\mu](X|Y) = \log \frac{1}{\mathscr{V}[\mu](X|Y)}$$

[3] Here, we adopt the trace based QIF formalization of [22].

where

$$\begin{array}{rcl} \mathscr{V}[\mu](X) & = & \max_{x\in\mathbb{X}} \mu(X=x) \\ \mathscr{V}[\mu](X|Y=y) & = & \max_{x\in\mathbb{X}} \mu(X=x|Y=y) \\ \mathscr{V}[\mu](X|Y) & = & \sum_{y\in\mathbb{Y}} \mu(Y=y)\mathscr{V}[\mu](X|Y=y) \end{array}$$

Intuitively, $\mathscr{V}[\mu](X)$ represents the highest probability that an attacker guesses X in a single try. We now define the min-entropy-based definition of QIF.

Definition 2.4 (Min-Entropy-based QIF [28]) *Let M be a program with a high security input H, and a low security output trace O. Let μ be a distribution over H. Then, the min-entropy-based quantitative information flow is defined*

$$ME[\mu](M) = \mathscr{H}_\infty[\mu](H) - \mathscr{H}_\infty[\mu](H|O)$$

Computing the min-entropy based quantitative information flow for our running example programs M_1 and M_2 from Section 1 with the uniform distribution, we obtain,

$$ME[U](M_1) = \mathscr{H}_\infty[U](H) - \mathscr{H}_\infty[U](H|O) = \log 4 - \log 2 = 1$$

$$ME[U](M_2) = \mathscr{H}_\infty[U](H) - \mathscr{H}_\infty[U](H|O) = \log 4 - \log 1 = 2$$

Again, we have that $ME[U](M_1) \leq ME[U](M_2)$ and $ME[U](M_2) \not\leq ME[U](M_1)$, and so M_2 is deemed less secure than M_1.

The third definition of quantitative information flow treated in this paper is the one based on the guessing entropy [23], that has also recently been proposed in literature [17, 3].

Definition 2.5 (Guessing Entropy) *Let X and Y be random variables, and μ be an associated probability distribution. Then, the guessing entropy of X is defined*

$$\mathscr{G}[\mu](X) = \sum_{1\leq i\leq m} i \times \mu(X=x_i)$$

where $\{x_1, x_2, \ldots, x_m\} = \mathbb{X}$ and $\forall i, j. i \leq j \Rightarrow \mu(X=x_i) \geq \mu(X=x_j)$.

The conditional guessing entropy of X given Y is defined

$$\mathscr{G}[\mu](X|Y) = \sum_{y\in\mathbb{Y}} \mu(Y=y) \sum_{1\leq i\leq m} i \times \mu(X=x_i|Y=y)$$

where $\{x_1, x_2, \ldots, x_m\} = \mathbb{X}$ and $\forall i, j. i \leq j \Rightarrow \mu(X=x_i|Y=y) \geq \mu(X=x_j|Y=y)$.

Intuitively, $\mathscr{G}[\mu](X)$ represents the average number of times required for the attacker to guess the value of X. We now define the guessing-entropy-based quantitative information flow.

Definition 2.6 (Guessing-Entropy-based QIF [17, 3]) *Let M be a program with a high security input H, and a low security output trace O. Let μ be a distribution over H. Then, the guessing-entropy-based quantitative information flow is defined*

$$GE[\mu](M) = \mathscr{G}[\mu](H) - \mathscr{G}[\mu](H|O)$$

We test GE on the running example from Section 1 by calculating the quantities for the programs M_1 and M_2 with the uniform distribution.

$$GE[U](M_1) = \mathscr{G}[U](H) - \mathscr{G}[U](H|O) = \tfrac{5}{2} - \tfrac{7}{4} = 0.75$$

$$GE[U](M_2) = \mathscr{G}[U](H) - \mathscr{G}[U](H|O) = \tfrac{5}{2} - 1 = 1.5$$

Therefore, we again have that $GE[U](M_1) \leq GE[U](M_2)$ and $GE[U](M_2) \not\leq GE[U](M_1)$, and so M_2 is considered less secure than M_1, even with the guessing-entropy based definition with the uniform distribution.

2.2 Bounding Problems

We introduce the bounding problems of quantitative information flow that we classify. First, we define the QIF upper-bounding problems. Upper-bounding problems are defined as follows: Given a program M and a rational number q, decide if the information flow of M is less than or equal to q.

$$\mathscr{U}_{SE} = \{(M,q) \mid SE[U](M) \leq q\}$$
$$\mathscr{U}_{ME} = \{(M,q) \mid ME[U](M) \leq q\}$$
$$\mathscr{U}_{GE} = \{(M,q) \mid GE[U](M) \leq q\}$$

Recall that U denotes the uniform distribution.

Next, we define lower-bounding problems. Lower-bounding problems are defined as follows: Given a program M and a rational number q, decide if the information flow of M is greater than q.

$$\mathscr{L}_{SE} = \{(M,q) \mid SE[U](M) > q\}$$
$$\mathscr{L}_{ME} = \{(M,q) \mid ME[U](M) > q\}$$
$$\mathscr{L}_{GE} = \{(M,q) \mid GE[U](M) > q\}$$

2.3 Non Interference

We recall the notion of non-interference, which, intuitively, says that the program leaks no information.

Definition 2.7 (Non-intereference [11, 15]) *A program M is said to be non-interfering iff for any $h,h' \in \mathbb{H}$, $M(h) = M(h')$.*

Non-interference is known to be a special case of bounding problems that tests against 0.

Theorem 2.8 ([7, 31]) *1.) M is non-interfering iff $(M,0) \in \mathscr{U}_{SE}$. 2.) M is non-interfering iff $(M,0) \in \mathscr{U}_{ME}$. 3.) M is non-interfering iff $(M,0) \in \mathscr{U}_{GE}$.*

3 Liveness Hyperproperties

Clarkson and Schneider have proposed the notion of hyperproperties [10].

Definition 3.1 (Hyperproperties [10]) *We say that P is a hyperproperty if $P \subseteq \mathscr{P}(\Psi_{\mathtt{inf}})$ where $\Psi_{\mathtt{inf}}$ is the set of all infinite traces, and $\mathscr{P}(X)$ denote the powerset of X.*

Note that hyperproperties are sets of trace sets. As such, they are more suitable for classifying information properties than the classical trace properties which are sets of traces. For example, non-interference is not a trace property but a hyperproperty.

Clarkson and Schneider have identified a subclass of hyperproperties, called liveness hyperproperties, as a generalization of liveness properties. Intuitively, a liveness hyperproperty is a property that can not be refuted by a finite set of finite traces. That is, if P is a liveness hyperproperty, then for any finite set of finite traces T, there exists a set of traces that contains T and satisfies P. Formally, let *Obs* be the set of finite sets of finite traces, and *Prop* be the set of sets of infinite traces (i.e., hyperproperties), that is,

$$\begin{array}{rcl} \mathit{Obs} & = & \mathscr{P}^{\mathtt{fin}}(\Psi_{\mathtt{fin}}) \\ \mathit{Prop} & = & \mathscr{P}(\Psi_{\mathtt{inf}}) \end{array}$$

(Here, $\mathscr{P}^{\mathtt{fin}}(X)$ denotes the finite subsets of X, $\Psi_{\mathtt{fin}}$ denotes the set of finite traces.) Let $\leq$ be the relation over $\mathit{Obs} \times \mathit{Prop}$ such that

$$S \leq T \quad \text{iff} \quad \forall t \in S. \exists t'. t \circ t' \in T$$

where $t \circ t'$ is the sequential composition of t and t'. Then,

Definition 3.2 (Liveness Hyperproperties [10]) *We say that a hyperproperty P is a liveness hyperproperty if for any set of traces $S \in Obs$, there exists a set of traces $S' \in Prop$ such that $S \leq S'$ and $S' \in P$.*

Now, we state the first main result of the paper: the lower-bounding problems are liveness hyperproperties.[4]

Theorem 3.3 *$\mathscr{L}_{SE}$, $\mathscr{L}_{ME}$, and $\mathscr{L}_{GE}$ are liveness hyperproperties.*[5]

The proof follows from the fact that, for any program M, there exists a program M' containing all the observations of M and has an arbitrary large information flow.[6]

We show that the upper-bounding problem for Shannon-entropy based quantitative information flow is also a liveness hyperproperty.

Theorem 3.4 *$\mathscr{U}_{SE}$ is a liveness hyperproperty.*

The theorem follows from the fact that we can lower the amount of the information flow by adding traces that have the same output trace. Therefore, for any program M, there exists M' having more observation than M such that $SE[U](M') \leq q$.

3.1 Observable Hyperproperties

Clarkson and Schneider [10] have identified a class of hyperproperties, called *observable hyperproperties*, to generalize the notion of observable properties [1] to sets of traces.[7]

Definition 3.5 (Observable Hyperproperties [10]) *We say that P is a observable hyperproperty if for any set of traces $S \in P$, there exists a set of traces $T \in Obs$ such that $T \leq S$, and for any set of traces $S' \in Prop$, $T \leq S' \Rightarrow S' \in P$.*

We call T in the above definition an *evidence*.

Intuitively, observable hyperproperty is a property that can be verified by observing a finite set of finite traces. We prove a relationship between observable hyperproperties and liveness hyperproperties.

Theorem 3.6 *Every non-empty observable hyperproperty is a liveness hyperproperty.*

Proof: Let P be a non-empty observable hyperproperty. It must be the case that there exists a set of traces $M \in P$. Then, there exists $T \in Obs$ such that $T \leq M$ and $\forall M' \in Prop. T \leq M' \Rightarrow M' \in P$. For any set of traces $S \in Obs$, there exists $M' \in Prop$ such that $S \leq M'$. Then, we have $M \cup M' \in P$, because $T \leq M \cup M'$. Therefore, P is a liveness hyperproperty. □

We note that the empty set is not a liveness hyperproperty but an observable hyperproperty.

We show that lower-bounding problems for min-entropy and guessing-entropy are observable hyperproperties.

Theorem 3.7 *$\mathscr{L}_{ME}$ is an observable hyperproperty.*

Theorem 3.8 *$\mathscr{L}_{GE}$ is an observable hyperproperty.*

[4] We implicitly extend the notion of hyperproperties to classify hyperproperties that take programs and rational numbers. See [31].

[5] More precisely, we prove that they are liveness hyperproperties for deterministic systems [10], because we restrict all programs to deterministic programs. For sake of simplicity, we omit such annotations.

[6] Here, we assume that the input domains are not bounded. Therefore, we can construct a program that leaks more high-security inputs by enlarging the input domain. Hyperproperty classifications of bounding problems with bounded domains appear in Section 5.1.

[7] Roughly, an observable property is a set of traces having a finite evidence prefix such that any trace having the prefix is also in the set.

Theorem 3.7 follows from the fact that, if $(M,q) \in \mathscr{L}_{ME}$, then M contains an evidence of $\mathscr{L}_{ME}$. This follows from the fact that when a program M' contains at least as much observation as M, $ME[U](M) \leq ME[U](M')$ (cf. Lemma 3.15). Theorem 3.8 is proven in a similar manner.

We show that neither of the bounding problems for Shannon-entropy are observable hyperproperties.

Theorem 3.9 *Neither $\mathscr{U}_{SE}$ nor $\mathscr{L}_{SE}$ is an observable hyperproperty.*

We give the intuition of the proof for $\mathscr{U}_{SE}$. Suppose $SE[U](M) \leq q$. M does not provide an evidence of $SE[U](M) \leq q$, because for any potential evidence, we can raise the amount of the information flow by adding traces that have disjoint output traces. The result for $\mathscr{L}_{SE}$ is shown in a similar manner.

It is interesting to note that the bounding problems of *SE* can only be classified as general liveness hyperproperties (cf. Theorem 3.3 and 3.4) even though *SE* is often the preferred definition of QIF in practice [13, 8, 21]. This suggests that approximation techniques may be necessary for checking and inferring Shannon-entropy-based QIF.

3.2 K-Observable Hyperproperties

We define k-observable hyperproperty that refines the notion of observable hyperproperties. Informally, a k-observable hyperproperty is a hyperproperty that can be verified by observing k finite traces.

Definition 3.10 (K-Observable Hyperproperties) *We say that a hyperproperty P is a k-observable hyperproperty if for any set of traces $S \in P$, there exists $T \in Obs$ such that $T \leq S$, $|T| \leq k$, and for any set of traces $S' \in Prop$, $T \leq S' \Rightarrow S' \in P$.*

Clearly, any k-observable hyperproperty is an observable hyperproperty.

We note that k-observable hyperproperties can be reduced to 1-observable hyperproperties by a simple program transformation called *self composition* [4, 12].

Definition 3.11 (Parallel Self Composition [10]) *Parallel self composition of S is defined as follows.*

$$S \times S = \{(s[0],s'[0]);(s[1],s'[1]);(s[2],s'[2]);\cdots \mid s,s' \in S\}$$

where $s[i]$ denotes the ith element of s.

Then, a k-product parallel self composition (simply self composition henceforth) is defined as S^k.

Theorem 3.12 *Every k-observable hyperproperty can be reduced to a 1-observable hyperproperty via a k-product self composition.*

As an example, consider the following hyperproperty. The hyperproperty is the set of programs that return 1 and 2 for some inputs. Intuitively, the hyperproperty expresses two good things happen (programs return 1 and 2) for programs.

$$\{M \mid \exists h,h'.M(h) = 1 \wedge M(h') = 2\}$$

This is a 2-observable hyperproperty as any program containing two traces, one having 1 as the output and the other having 2 as the output, satisfies it.

We can check the above property by self composition. (Here, || denotes a parallel composition.)

$$M'(H,H') \quad \equiv \quad O := M(H) \,||\, O' := M(H') \,||\, \mathsf{assert}(\neg(\mathsf{O} = 1 \wedge \mathsf{O}' = 2))$$

Clearly, M satisfies the property iff the assertion failure is reachable in the above program, that is, iff the predicate $O = 1 \wedge O' = 2$ holds for some inputs H,H'. (Note that, for convenience, we take an assertion failure to be a "good thing".)

We show that neither the lower-bounding problem for min-entropy nor the lower-bounding problem for guessing-entropy is a k-observable hyperproperty for any k.

Theorem 3.13 *Neither $\mathscr{L}_{ME}$ nor $\mathscr{L}_{GE}$ is a k-observable property for any k.*

However, if we let q be a constant, then we obtain different results. First, we show that the lower-bounding problem for min-entropy-based quantitative information flow under a constant bound q, is a $\lfloor 2^q \rfloor + 1$-observable hyperproperty.

Theorem 3.14 *Let q be a constant. Then, $\mathscr{L}_{ME}$ is a $\lfloor 2^q \rfloor + 1$-observable hyperproperty.*

The theorem follows from Lemma 3.15 below which states that min-entropy based quantitative information flow under the uniform distribution coincides with the logarithm of the number of output traces. That is, $(M,q) \in \mathscr{L}_{ME}$ iff there is an evidence in M containing $\lfloor 2^q \rfloor + 1$ disjoint outputs.

Lemma 3.15 ([28]) $ME[U](M) = \log |\{o \mid \exists h.M(h) = o\}|$

Next, we show that the lower-bounding problem for guessing-entropy-based quantitative information flow under a constant bound q is a $\lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$-observable hyperproperty.

Theorem 3.16 *Let q be a constant. Then, $\mathscr{L}_{GE}$ is a $\lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$-observable hyperproperty.*

The proof of the theorem is similar to that of Theorem 3.14, in that the size of the evidence set can be computed from the bound q.

3.3 Computational Complexities

We prove computational complexities of $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ by utilizing their hyperproperty classifications. Following previous work [32, 31, 6], we focus on boolean programs.

First, we introduce the syntax of boolean programs. The semantics of boolean programs is standard and is deferred to Appendix (Figure 4). We call boolean programs without while statements *loop-free* boolean programs.

$$\begin{array}{rcl} M & ::= & x := \psi \mid M_0; M_1 \mid \mathsf{if}\ \psi\ \mathsf{then}\ M_0\ \mathsf{else}\ M_1 \mid \mathsf{while}\ \psi\ \mathsf{do}\ M \mid \texttt{skip} \\ \phi, \psi & ::= & \mathsf{true} \mid x \mid \phi \wedge \psi \mid \neg \phi \end{array}$$

Figure 1: The syntax of boolean programs

In this paper, we are interested in the computational complexity with respect to the syntactic size of the input program (i.e., "implicit state complexity", as opposed to [6] which studies complexity over programs represented as explicit states).

We show that the lower-bounding problems for min-entropy and guessing-entropy are *PP*-hard.

Theorem 3.17 *$\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ for loop-free boolean programs are PP-hard.*

The theorem is proven by a reduction from MAJSAT, which is a *PP*-hard problem. *PP* is the set of decision problems solvable by a polynomial-time nondeterministic Turing machine which accepts the input iff more than half of the computation paths accept. MAJSAT is the problem of deciding, given a boolean formula ϕ over variables $\overrightarrow{x}$, if there are more than $2^{|\overrightarrow{x}|-1}$ satisfying assignments to ϕ.

Next, we show that if q be a constant, the upper-bounding problems for min-entropy and guessing-entropy become *NP*-complete.

Theorem 3.18 *Let q be a constant. Then, $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are NP-complete for loop-free boolean programs.*

NP-hardness is proven by a reduction from SAT, which is a *NP*-complete problem. The proof that $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ for a constant q are in *NP* follows from the fact that $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are k-observable hyperproperties for some k. We give the proof intuition for $\mathscr{L}_{ME}$. Recall that k-observable hyperproperties can be reduced to 1-observable hyperproperties via self composition. Consequently, it is possible to decide if the information flow of a given program M is greater than q by checking if the predicate of the assert statement is violated for some inputs in the following program.

$$\begin{array}{l} M'(H_1, H_2, \ldots, H_n) \equiv \\ \quad O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n); \\ \quad \mathsf{assert}(\bigvee_{i,j \in \{1,\ldots,n\}} (O_i = O_j \wedge i \neq j)) \end{array}$$

where $n = \lfloor 2^q \rfloor + 1$. Let ϕ be the weakest precondition of $O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n)$ with respect to the post condition $\bigvee_{i,j \in \{1,\ldots,n\}} (O_i = O_j \wedge i \neq j)$. Then, $ME[U](M) > q$ iff $\neg\phi$ is satisfiable. Because a weakest precondition of a loop-free boolean program is a polynomial size boolean formula over the boolean variables representing the inputs[8], deciding $ME[U](M) > q$ is reducible to SAT.

For boolean programs (with loops), $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are *PSPACE*-complete, and $\mathscr{L}_{SE}$ is *PSPACE*-hard (the tight upper-bound is open for $\mathscr{L}_{SE}$).

Theorem 3.19 *$\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are PSPACE-complete for boolean programs.*

Theorem 3.20 *$\mathscr{L}_{SE}$ is PSPACE-hard for boolean programs.*

4 Safety Hyperproperties

Clarkson and Schneider [10] have proposed safety hyperproperties, a subclass of hyperproperties, as a generalization of safety properties. Intuitively, a safety hyperproperty is a hyperproperty that can be refuted by observing a finite set of finite traces.

Definition 4.1 (Safety Hyperproperties [10]) *We say that a hyperproperty P is a safety hyperproperty if for any set of traces $S \notin P$, there exists a set of traces $T \in Obs$ such that $T \leq S$, and $\forall S' \in Prop.T \leq S' \Rightarrow S' \notin P$.*

We classify some upper-bounding problems as safety hyperproperties.

Theorem 4.2 *U_{ME} and U_{GE} are safety hyperproperties.*

Next, we review the definition of k-safety hyperproperties [10], which refines the notion of safety hyperproperties. Informally, a k-safety hyperproperty is a hyperproperty which can be refuted by observing k number of finite traces.

Definition 4.3 (K-Safety Hyperproperties [10]) *We say that a hyperproperty P is a k-safety property if for any set of traces $S \notin P$, there exists a set of traces $T \in Obs$ such that $T \leq S$, $|T| \leq k$, and $\forall S' \in Prop.T \leq S' \Rightarrow S' \notin P$.*

Note that 1-safety hyperproperty is just the standard safety property, that is, a property that can be refuted by observing a finite execution trace. The notion of k-safety hyperproperties first came into limelight when it was noticed that non-interference is a 2-safety hyperproperty, but not a 1-safety hyperproperty [29].

A k-safety hyperproperty can be reduced to a 1-safety hyperproperty by self composition [4, 12].

[8] For loop-free boolean programs, a weakest precondition can be constructed in polynomial time [14, 20].

Theorem 4.4 ([10]) *k-safety hyperproperty can be reduced to* 1*-safety hyperproperty by self composition.*

We have shown in our previous work that $\mathscr{U}_{ME}$ and $\mathscr{U}_{GE}$ are k-safety hyperproperties when the bound q is fixed to a constant.

Theorem 4.5 ([31]) *Let q be a constant. $\mathscr{U}_{ME}$ is a $\lfloor 2^q \rfloor + 1$-safety property.*

Theorem 4.6 ([31]) *Let q be a constant. $\mathscr{U}_{GE}$ is a $\lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$-safety property.*

The only hyperproperty that is both a safety hyperproperty and a liveness hyperproperty is $\mathscr{P}(\Psi_{\mathtt{inf}})$, that is, the set of all traces [10]. Consequently, neither $\mathscr{U}_{ME}$ nor $\mathscr{U}_{GE}$ is a liveness hyperproperty.

We have also shown in the previous work that the upper-bounding problem for Shannon-entropy based quantitative information flow is not a k-safety hyperproperty, even when q is a constant.

Theorem 4.7 ([31]) *Let q be a constant. $\mathscr{U}_{SE}$ is not a k-safety property for any $k > 0$.*

4.1 Computational Complexities

We prove computational complexities of upper-bounding problems by utilizing their hyperproperty classifications. As in Section 3.3, we focus on boolean programs.

First, we show that when q is a constant, U_{ME} and U_{GE} are *coNP*-complete.

Theorem 4.8 *Let q be a constant. Then, $\mathscr{U}_{ME}$ and $\mathscr{U}_{GE}$ are coNP-complete for loop-free boolean programs.*

coNP-hardness follows from the fact that non-interference is *coNP*-hard [31]. The *coNP* part of the proof is similar to the *NP* part of Theorem 3.18, and uses the fact that $\mathscr{U}_{ME}$ is k-safety for a fixed q and uses self composition. By self composition, the upper-bounding problem can be reduced to a reachability problem (i.e., an assertion failure is unreachable for any input). To decide if $ME[U](M) \leq q$, we construct the following self-composed program M' from the given program M.

$$
\begin{array}{l}
M'(H_1, H_2, \ldots, H_n) \equiv \\
\quad O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n); \\
\quad \mathsf{assert}(\bigvee_{i,j \in \{1,\ldots,n\}} (O_i = O_j \wedge i \neq j))
\end{array}
$$

where $n = \lfloor 2^q \rfloor + 1$. Then, the weakest precondition of $O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n)$ with respect to the post condition $\bigvee_{i,j \in \{1,\ldots,n\}} (O_i = O_j \wedge i \neq j)$ is valid iff $ME[U](M) \leq q$. Because a weakest precondition of a loop-free boolean program is a polynomial size boolean formula, and the problem of deciding a given boolean formula is valid is a *coNP*-complete problem, $\mathscr{U}_{ME}$ is in *coNP*.

Like the lower-bounding problems $\mathscr{U}_{ME}$ and $\mathscr{U}_{GE}$ for boolean programs (with loops) are *PSPACE*-complete, and $\mathscr{U}_{SE}$ is *PSPACE*-hard.

Theorem 4.9 *$\mathscr{U}_{ME}$ and $\mathscr{U}_{GE}$ are PSPACE-complete for boolean programs.*

Theorem 4.10 *$\mathscr{U}_{SE}$ is PSPACE-hard for boolean programs.*

5 Discussion

5.1 Bounding Domains

The notion of hyperproperty is defined over all programs regardless of their size. (For example, non-interference is a 2-safety property for all programs and reachability is a safety property for all programs.) But, it is easy to show that the lower bounding problems would become "k-observable" hyperproperties if we constrained and bounded the input domains because then the size of the semantics (i.e., the number of traces) of such programs would be bounded by $|\mathbb{H}|$ (and upper bounding problems would become "k-safety" hyperproperties [31]). In this case, the problems are trivially $|\mathbb{H}|$-observable hyperproperties. However, these bounds are high for all but very small domains, and are unlikely to lead to a practical verification method.

5.2 Observable Hyperproperties and Observable Properties

As remarked in [10], observable hyperproperties generalize the notion of observable properties [1]. It can be shown that there exists a non-empty observable property that is not a liveness property (e.g., the set of all traces that starts with σ). In contrast, Theorem 3.6 states that every non-empty observable hyperproperty is also a liveness hyperproperty. Intuitively, this follows because the hyperproperty extension relation $\leq$ allows the right-hand side to contain traces that does not appear in the left-hand side. Therefore, for any $T \in \mathit{Obs}$, there exists $T' \in \mathit{Prop}$ that contains T and an evidence of the observable hyperproperty.

5.3 Maximum of QIF over Distribution

Researchers have studied the maximum of QIF over the distribution. For example, *channel capacity* [24, 22, 26] is the maximum of the Shannon-entropy based quantitative information flow over the distribution (i.e., $\max_\mu SE[\mu]$). Smith [28] showed that for any program without low-security inputs, the channel capacity is equal to the min-entropy-based quantitative information flow, that is, $\max_\mu SE[\mu] = ME[U]$. Therefore, we obtain the same hyperproperty classifications and complexity results for channel capacity as $ME[U]$.

Min-entropy channel capacity and *guessing-entropy channel capacity* are respectively the maximums of min-entropy based and guessing-entropy based QIF over distributions (i.e., $\max_\mu ME[\mu]$ and $\max_\mu GE[\mu]$). It has been shown that $\max_\mu ME[\mu] = ME[U]$ [5, 19] and $\max_\mu GE[\mu] = GE[U]$ [33], that is, they attain their maximums when the distributions are uniform. Therefore, they have the same hyperproperty classifications and complexities as $ME[U]$ and $GE[U]$, which we have already analyzed in this paper.

6 Related Work

Černý et al. [6] have investigated the computational complexity of Shannon-entropy based QIF. Formally, they have defined a Shannon-entropy based QIF for interactive boolean programs, and showed that the explicit-state computational complexity of their lower-bounding problem is *PSPACE*-complete. In contrast, this paper's complexity results are "implicit" complexity results of bounding problems of boolean programs (i.e., complexity relative to the syntactic size of the input) some of which are obtained by utilizing their hyperproperties classifications.

Clarkson and Schneider [10] have classified quantitative information flow problems via hyperproperties. Namely, they have shown that the problem of deciding if the channel capacity of a given program is q, is a liveness hyperproperty. And, they have shown that an upper-bounding problem for the *belief*-based QIF [9] is a safety hyperproperty. (It is possible to refine their result to show that their problem for deterministic programs is actually equivalent to non-interference, and therefore, is a 2-safety hyperproperty [33].)

7 Conclusion

We have related the upper and lower bounding problems of quantitative information flow, for various information theoretic definitions proposed in literature, to Clarkson and Schneider's hyperproperties. Hyperproperties generalize the classical trace properties, and are thought to be more suitable for classifying information flow properties as they are relations over sets of program traces. Our results confirm this by giving a fine-grained classification and showing that it gives insights into the complexity of the QIF bounding problems. One of the contributions is a new class of hyperproperties: *k-observable* hyperproperty. We have shown that k-observable hyperproperties are amenable to verification via self composition.

References

[1] Samson Abramsky (1991): *Domain Theory in Logical Form*. *Ann. Pure Appl. Logic* 51(1-2), pp. 1–77. Available at `http://dx.doi.org/10.1016/0168-0072(91)90065-T`.

[2] Michael Backes, Matthias Berg & Boris Köpf (2011): *Non-uniform distributions in quantitative information-flow*. In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, ACM, New York, NY, USA, pp. 367–375, doi:http://doi.acm.org/10.1145/1966913.1966960. Available at `http://doi.acm.org/10.1145/1966913.1966960`.

[3] Michael Backes, Boris Köpf & Andrey Rybalchenko (2009): *Automatic Discovery and Quantification of Information Leaks*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 141–153. Available at `http://dx.doi.org/10.1109/SP.2009.18`.

[4] Gilles Barthe, Pedro R. D'Argenio & Tamara Rezk (2004): *Secure Information Flow by Self-Composition*. In: *CSFW*, IEEE Computer Society, pp. 100–114. Available at `http://doi.ieeecomputersociety.org/10.1109/CSFW.2004.17`.

[5] Christelle Braun, Konstantinos Chatzikokolakis & Catuscia Palamidessi (2009): *Quantitative Notions of Leakage for One-try Attacks*. *Electr. Notes Theor. Comput. Sci.* 249, pp. 75–91. Available at `http://dx.doi.org/10.1016/j.entcs.2009.07.085`.

[6] Pavol Černý, Krishnendu Chatterjee & Thomas A. Henzinger (2011): *The Complexity of Quantitative Information Flow Problems*. In: *CSF*, IEEE Computer Society, pp. 205–217. Available at `http://doi.ieeecomputersociety.org/10.1109/CSF.2011.21`.

[7] David Clark, Sebastian Hunt & Pasquale Malacaria (2005): *Quantified Interference for a While Language*. *Electr. Notes Theor. Comput. Sci.* 112, pp. 149–166. Available at `http://dx.doi.org/10.1016/j.entcs.2004.01.018`.

[8] David Clark, Sebastian Hunt & Pasquale Malacaria (2007): *A static analysis for quantifying information flow in a simple imperative language*. *J. Comput. Secur.* 15, pp. 321–371. Available at `http://dl.acm.org/citation.cfm?id=1370628.1370629`.

[9] Michael R. Clarkson, Andrew C. Myers & Fred B. Schneider (2005): *Belief in Information Flow*. In: *CSFW*, IEEE Computer Society, pp. 31–45. Available at `http://dx.doi.org/10.1109/CSFW.2005.10`.

[10] Michael R. Clarkson & Fred B. Schneider (2010): *Hyperproperties*. Journal of Computer Security 18(6), pp. 1157–1210. Available at http://dx.doi.org/10.3233/JCS-2009-0393.

[11] Ellis S. Cohen (1977): *Information Transmission in Computational Systems*. In: *SOSP*, pp. 133–139. Available at http://doi.acm.org/10.1145/800214.806556.

[12] Ádám Darvas, Reiner Hähnle & David Sands (2005): *A Theorem Proving Approach to Analysis of Secure Information Flow*. In Dieter Hutter & Markus Ullmann, editors: *SPC*, Lecture Notes in Computer Science 3450, Springer, pp. 193–209. Available at http://dx.doi.org/10.1007/978-3-540-32004-3_20.

[13] Dorothy Elizabeth Robling Denning (1982): *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[14] Cormac Flanagan & James B. Saxe (2001): *Avoiding exponential explosion: generating compact verification conditions*. In: *POPL*, pp. 193–205. Available at http://doi.acm.org/10.1145/360204.360220.

[15] Joseph A. Goguen & José Meseguer (1982): *Security Policies and Security Models*. In: *IEEE Symposium on Security and Privacy*, pp. 11–20.

[16] Jonathan Heusser & Pasquale Malacaria (2009): *Applied Quantitative Information Flow and Statistical Databases*. In Pierpaolo Degano & Joshua D. Guttman, editors: *Formal Aspects in Security and Trust*, Lecture Notes in Computer Science 5983, Springer, pp. 96–110. Available at http://dx.doi.org/10.1007/978-3-642-12459-4_8.

[17] Boris Köpf & David A. Basin (2007): *An information-theoretic model for adaptive side-channel attacks*. In Peng Ning, Sabrina De Capitani di Vimercati & Paul F. Syverson, editors: *ACM Conference on Computer and Communications Security*, ACM, pp. 286–296. Available at http://doi.acm.org/10.1145/1315245.1315282.

[18] Boris Köpf & Andrey Rybalchenko (2010): *Approximation and Randomization for Quantitative Information-Flow Analysis*. In: *CSF*, IEEE Computer Society, pp. 3–14. Available at http://doi.ieeecomputersociety.org/10.1109/CSF.2010.8.

[19] Boris Köpf & Geoffrey Smith (2010): *Vulnerability Bounds and Leakage Resilience of Blinded Cryptography under Timing Attacks*. In: *CSF*, IEEE Computer Society, pp. 44–56. Available at http://doi.ieeecomputersociety.org/10.1109/CSF.2010.11.

[20] K. Rustan M. Leino (2005): *Efficient weakest preconditions*. Inf. Process. Lett. 93(6), pp. 281–288. Available at http://dx.doi.org/10.1016/j.ipl.2004.10.015.

[21] Pasquale Malacaria (2007): *Assessing security threats of looping constructs*. In Martin Hofmann & Matthias Felleisen, editors: *POPL*, ACM, pp. 225–235. Available at http://doi.acm.org/10.1145/1190216.1190251.

[22] Pasquale Malacaria & Han Chen (2008): *Lagrange multipliers and maximum information leakage in different observational models*. In Úlfar Erlingsson & Marco Pistoia, editors: *PLAS*, ACM, pp. 135–146. Available at http://doi.acm.org/10.1145/1375696.1375713.

[23] James L. Massey (1994): *Guessing and Entropy*. In: *ISIT '94: Proceedings of the 1994 IEEE International Symposium on Information Theory*, p. 204. Available at http://dx.doi.org/10.1109/ISIT.1994.394764.

[24] Stephen McCamant & Michael D. Ernst (2008): *Quantitative information flow as network flow capacity*. In Rajiv Gupta & Saman P. Amarasinghe, editors: *PLDI*, ACM, pp. 193–205. Available at http://doi.acm.org/10.1145/1375581.1375606.

[25] David A. Naumann (2006): *From Coupling Relations to Mated Invariants for Checking Information Flow*. In Dieter Gollmann, Jan Meier & Andrei Sabelfeld, editors: *ESORICS*, Lecture Notes in Computer Science 4189, Springer, pp. 279–296. Available at http://dx.doi.org/10.1007/11863908_18.

[26] James Newsome, Stephen McCamant & Dawn Song (2009): *Measuring channel capacity to distinguish undue influence*. In Stephen Chong & David A. Naumann, editors: *PLAS*, ACM, pp. 73–85. Available at http://doi.acm.org/10.1145/1554339.1554349.

[27] Claude Shannon (1948): *A Mathematical Theory of Communication*. *Bell System Technical Journal* 27, pp. 379–423, 623–656. Available at http://doi.acm.org/10.1145/584091.584093.

[28] Geoffrey Smith (2009): *On the Foundations of Quantitative Information Flow*. In Luca de Alfaro, editor: *FOSSACS*, *Lecture Notes in Computer Science* 5504, Springer, pp. 288–302. Available at http://dx.doi.org/10.1007/978-3-642-00596-1_21.

[29] Tachio Terauchi & Alexander Aiken (2005): *Secure Information Flow as a Safety Problem*. In Chris Hankin & Igor Siveroni, editors: *SAS*, *Lecture Notes in Computer Science* 3672, Springer, pp. 352–367. Available at http://dx.doi.org/10.1007/11547662_24.

[30] Hiroshi Unno, Naoki Kobayashi & Akinori Yonezawa (2006): *Combining type-based analysis and model checking for finding counterexamples against non-interference*. In Vugranam C. Sreedhar & Steve Zdancewic, editors: *PLAS*, ACM, pp. 17–26. Available at http://doi.acm.org/10.1145/1134744.1134750.

[31] Hirotoshi Yasuoka & Tachio Terauchi (2010): *On Bounding Problems of Quantitative Information Flow*. In Dimitris Gritzalis, Bart Preneel & Marianthi Theoharidou, editors: *ESORICS*, *Lecture Notes in Computer Science* 6345, Springer, pp. 357–372. Available at http://dx.doi.org/10.1007/978-3-642-15497-3_22.

[32] Hirotoshi Yasuoka & Tachio Terauchi (2010): *Quantitative Information Flow - Verification Hardness and Possibilities*. In: *CSF*, IEEE Computer Society, pp. 15–27. Available at http://doi.ieeecomputersociety.org/10.1109/CSF.2010.9.

[33] Hirotoshi Yasuoka & Tachio Terauchi (2011): *On Bounding Problems of Quantitative Information Flow (Extended version)*. *Journal of Computer Security* 19(6), pp. 1029–1082. Available at http://dx.doi.org/10.3233/JCS-2011-0437.

A Appendix

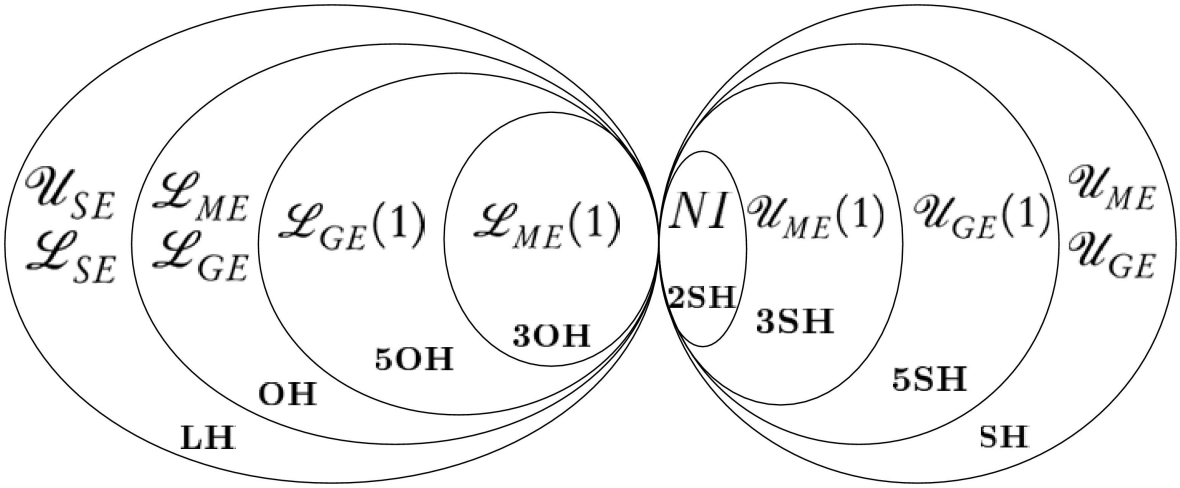

Figure 2: A summary of hyperproperty classifications

Fig 2 shows a summary of hyperproperty classifications.

- **LH**: the set of all liveness hyperproperties.
- **OH**: the set of all non-empty observable hyperproperties.

- **3OH**: the set of all non-empty 3-observable hyperproperties.
- **5OH**: the set of all non-empty 5-observable hyperproperties.
- **SH**: the set of all safety hyperproperties.
- **2SH**: the set of all 2-safety hyperproperties.
- **3SH**: the set of all 3-safety hyperproperties.
- **5SH**: the set of all 5-safety hyperproperties.
- $\mathscr{L}_{ME}(1)$: $\mathscr{L}_{ME}$ under the constant bound 1.
- $\mathscr{L}_{GE}(1)$: $\mathscr{L}_{GE}$ under the constant bound 1.
- $\mathscr{U}_{ME}(1)$: $\mathscr{U}_{ME}$ under the constant bound 1.
- $\mathscr{U}_{GE}(1)$: $\mathscr{U}_{GE}$ under the constant bound 1.
- *NI*: the set of all non-interfering programs.

In general, for a constant bound q, $\mathscr{L}_{ME}(q)$ is in the class $\lfloor 2^q \rfloor + 1$-**OH** and $\mathscr{L}_{GE}(q)$ is in the class $\lfloor \frac{(\lfloor q \rfloor+1)^2}{\lfloor q \rfloor+1-q} \rfloor + 1$-**OH**, and such classes are all contained in **OH**. Likewise, for a constant bound q, $\mathscr{U}_{ME}(q)$ is in the class $\lfloor 2^q \rfloor + 1$-**SH** and $\mathscr{U}_{GE}(q)$ is in the class $\lfloor \frac{(\lfloor q \rfloor+1)^2}{\lfloor q \rfloor+1-q} \rfloor + 1$-**SH**, with such classes all contained in **SH**.

A.1 Omitted Proof

Definition A.1

$$In(\mu, X, x) = |\{x' \in X \mid \mu(x') \geq \mu(x)\}|$$

Intuitively, $In(\mu, X, x)$ is the order of x defined in terms of μ.

Lemma A.2

$$\mathscr{G}[\mu](X) = \Sigma_{1 \leq i \leq |X|} i\mu(x_i) = \Sigma_{x \in X} In(\mu, X, x)\mu(x)$$

Proof: Trivial. □

Theorem 2.8. *1.) M is non-interfering iff* $(M, 0) \in \mathscr{U}_{SE}$. *2.) M is non-interfering iff* $(M, 0) \in \mathscr{U}_{ME}$. *3.) M is non-interfering iff* $(M, 0) \in \mathscr{U}_{GE}$.
Proof: Let $\mathbb{O} = \{M(h, \ell) \mid h \in \mathbb{H} \wedge \ell \in \mathbb{L}\}$.

- *SE*

  (See [7].)

- *ME*
  - $\Rightarrow$

    Suppose M is non-interfering. By definition, it suffices to show that

    $$\mathscr{V}[U](H) = \mathscr{V}[U](H|O)$$

    That is,

    $$\max_h U(h) = \sum_o U(o) \max_h U(h|o)$$

The fact that M is non-interfering implies that there exists o' such that for any h, $M(h) = o'$, and $U(h,o') = U(h)$. Therefore, we have

$$\begin{aligned}\textstyle\sum_o U(o)\max_h U(h|o) &= \max_h \frac{U(h,o')}{U(o')} \\ &= \max_h U(h)\end{aligned}$$

- $\Leftarrow$

  We prove the contraposition. Suppose M is interfering. That is, there exist h_1 and h_2 such that $M(h_1) \neq M(h_2)$. Let $o_1 = M(h_1)$ and $o_2 = M(h_2)$. We have

$$\begin{aligned}&\textstyle\sum_o U(o)\max_h U(h|o) \\ &\quad = \textstyle\sum_{o\in\mathbb{O}\setminus\{o_1,o_2\}} \max_h U(h,o) + \max_h U(h,o_1) + \max_h U(h,o_2)\end{aligned}$$

  And,

$$\max_h U(h) = \max_h U(h,o_1)$$

  Therefore, we have $ME[U](M) > 0$.

- *GE*

  - $\Rightarrow$

    Suppose M is non-interfering. By definition,

$$\begin{aligned}&GE[U](M) \\ &\quad = \textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ &\qquad - \textstyle\sum_o\sum_h In(\lambda h'.U(h',o),\mathbb{H},h)U(h,o) \\ &\quad = \textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ &\qquad - \textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ &\quad = 0\end{aligned}$$

    since for all h_x and o_x such that $U(h_x,o_x) > 0$, for any h'_x and $o' \in \mathbb{O}\setminus\{o_x\}$, $U(h'_x,o'_x) = 0$.

  - $\Leftarrow$

    We prove the contraposition. Suppose M is interfering. That is, there exist h_1 and h_2 such that $M(h_1) \neq M(h_2)$. Let $o_1 = M(h_1)$ and $o_2 = M(h_2)$. By the definition,

$$\begin{aligned}&GE[U](M) \\ &\quad = \textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ &\qquad - \textstyle\sum_o\sum_h In(\lambda h'.U(h',o),\mathbb{H},h)U(h,o) \\ &\quad = A + \textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ &\qquad - B - \textstyle\sum_o\sum_h In(\lambda h'.U(h',o),\mathbb{H},h)U(h,o)\end{aligned}$$

    where

$$\begin{aligned}A &= \textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ B &= \textstyle\sum_{o\in\mathbb{O}}\sum_h In(\lambda h'.U(h',o),\mathbb{H},h)U(h,o)\end{aligned}$$

    Trivially, we have $A \geq B$ and

$$\begin{aligned}&\textstyle\sum_h In(\lambda h'.U(h'),\mathbb{H},h)U(h) \\ &\qquad > \textstyle\sum_o\sum_h In(\lambda h'.U(h',o),\mathbb{H},h)U(h,o)\end{aligned}$$

    Therefore, we have $GE[U](M) > 0$.

□

Hereafter, we write sets of pairs of input and output trace to mean programs. That is, we write $(h,o) \in M$ iff $M(h) = o$.

Lemma A.3 *For any rational numbers q, for any traces $T \in Obs$, there exist M' and M'' such that $T \leq M'$, $T \leq M''$, $SE[U](M') \leq q$, and $SE[U](M'') > q$.*

Proof: Firstly, we prove that there exists M' such that $T \leq M'$ and $SE[U](M') \leq q$. Let M_T be a program such that $|M_T| = |T|$ and $T \leq M_T$. We construct the following programs.

$$M' = M_T \cup \{(h,o) \mid h \in \mathbb{H}'\}$$
$$M_1 = \{(h,o_h) \mid h \in \mathbb{H}\} \cup \{(h,o) \mid h \in \mathbb{H}'\}$$

where

- $\mathbb{H} = dom(M)$,
- $\mathbb{H}' \cap \mathbb{H} = \emptyset$,
- for any h and h' in $\mathbb{H}$, $h \neq h' \Rightarrow o_h \neq o_{h'}$,
- for any $h \in \mathbb{H}'$, $o \neq M(h)$, and
- for any $h' \in \mathbb{H}'$, $o \neq o_h$.

It follows that

$$SE[U](M') \leq SE[U](M_1)$$

And, we have

$$\begin{array}{rcll} SE[U](M_1) & = & \frac{|\mathbb{H}|}{|\mathbb{H}|+|\mathbb{H}'|}\log(|\mathbb{H}|+|\mathbb{H}'|) + \frac{|\mathbb{H}'|}{|\mathbb{H}|+|\mathbb{H}'|}\log\frac{|\mathbb{H}|+|\mathbb{H}'|}{|\mathbb{H}'|} \\ & \to & 0 & (|\mathbb{H}'| \to \infty) \end{array}$$

Therefore, there exists $\mathbb{H}'$ such that $SE[U](M') \leq q$.

Finally, we prove that there exists M'' such that $T \leq M''$ and $SE[U](M'') > q$. Let M_T be a program such that $|M_T| = |T|$ and $T \leq M_T$. We construct the following programs.

$$M'' = M_T \cup \{(h,o_h) \mid h \in \mathbb{H}'\}$$
$$M_1 = \{(h,o) \mid h \in \mathbb{H}\} \cup \{(h,o_h) \mid h \in \mathbb{H}'\}$$

where

- $\mathbb{H} = dom(M)$,
- $\mathbb{H}' \cap \mathbb{H} = \emptyset$,
- for any h and h' in $\mathbb{H}'$, $h \neq h' \Rightarrow o_h \neq o_{h'}$, and
- for any $h' \in \mathbb{H}'$, for any $h \in \mathbb{H}$, $o_{h'} \neq M(h)$

It follows that

$$SE[U](M'') \geq SE[U](M_1)$$

And, we have

$$\begin{array}{rcll} SE[U](M_1) & = & \frac{|\mathbb{H}'|}{|\mathbb{H}|+|\mathbb{H}'|}\log(|\mathbb{H}|+|\mathbb{H}'|) + \frac{|\mathbb{H}|}{|\mathbb{H}|+|\mathbb{H}'|}\log\frac{|\mathbb{H}|+|\mathbb{H}'|}{|\mathbb{H}|} \\ & \to & \infty & (|\mathbb{H}'| \to \infty) \end{array}$$

Therefore, there exists $\mathbb{H}'$ such that $SE[U](M'') > q$.
□

Theorem 3.3. *$\mathscr{L}_{SE}$, $\mathscr{L}_{ME}$, and $\mathscr{L}_{GE}$ are liveness hyperproperties.*
Proof: The fact that $\mathscr{L}_{SE}$ is a liveness hyperproperty follows from Lemma A.3. The results for $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ follow from Theorem 3.7, Theorem 3.8, and Theorem 3.6. □

Theorem 3.4. *$\mathscr{U}_{SE}$ is a liveness hyperproperty.*
Proof: Straightforward by Lemma A.3. □

Theorem 3.7. *$\mathscr{L}_{ME}$ is a observable hyperproperty.*
Proof: Let M be a program and q be a rational number such that $(M,q) \in \mathscr{L}_{ME}$. By the definition of the equivalence relation on outputs, there exists $T \in Obs$ such that $ME[U](T) = ME[U](M)$. By Lemma 3.15, for any programs M', if $T \leq M'$, then $ME[U](M') > q$. Therefore, $\mathscr{L}_{ME}$ is a observable hyperproperty. □

Lemma A.4 *Let M be a program. Then, we have $GE[U](M) = \frac{n}{2} - \frac{1}{2n}\sum_o |\mathbb{H}_o|^2$ where n is the number of inputs, and $\mathbb{H}_o = \{h \mid o = M(h)\}$.*

Proof: By the definition, we have

$$\begin{array}{ll} GE[U](M) & = \mathscr{G}[U](H) - \mathscr{G}[U](H|O) \\ & = \sum_h In(U,\mathbb{H},h)U(h) \\ & \quad - \sum_o U(o) \sum_h In(\lambda h'.U(h'|o),\mathbb{H}_o,h)U(h|o) \\ & = \frac{1}{n}\frac{1}{2}n(n+1) - \sum_o \frac{|\mathbb{H}_o|}{n}\frac{1}{2}\frac{1}{|\mathbb{H}_o|}|\mathbb{H}_o|(|\mathbb{H}_o|+1) \\ & = \frac{n}{2} + \frac{1}{2} - \frac{1}{2n}\sum_o |\mathbb{H}_o|^2 - \frac{1}{2} \\ & = \frac{n}{2} - \frac{1}{2n}\sum_o |\mathbb{H}_o|^2 \end{array}$$

□

Lemma A.5 *Let M and M' be programs such that $[\![M']\!] = [\![M]\!] \cup \{(h,o)\}$ and $h \notin dom([\![M]\!])$. Then, we have $GE[U](M) \leq GE[U](M')$.*

Proof: We prove $GE[U](M') - GE[U](M) \geq 0$. Let $n = |[\![M]\!]|$, $\mathbb{O} = ran([\![M]\!])$, $\mathbb{H} = dom(M)$, and $\mathbb{H}_o = dom(h \mid o = M(h))$.

By Lemma A.4, we have

$$\begin{array}{l} GE[U](M') - GE[U](M) \\ \quad = \frac{n+1}{2} - \frac{1}{2(n+1)}(B + (|\mathbb{H}_o|+1)^2) - \frac{n}{2} + \frac{1}{2n}(B + |\mathbb{H}_o|^2) \\ \quad = \frac{1}{2} + \frac{1}{2n(n+1)}(-n(B + (|\mathbb{H}_o|+1)^2) + (n+1)(B + |\mathbb{H}_o|^2)) \\ \quad = \frac{1}{2n(n+1)}(n(n+1) - n(|\mathbb{H}_o|+1)^2 + B + (n+1)(|\mathbb{H}_o|^2)) \\ \quad = \frac{1}{2n(n+1)}(n^2 - 2n|\mathbb{H}_o| + |\mathbb{H}_o|^2 + B) \\ \quad = \frac{1}{2n(n+1)}((n - |\mathbb{H}_o|)^2 + B) \\ \quad \geq 0 \end{array}$$

where $B = \sum_{o' \in \mathbb{O} \setminus \{o\}} |\mathbb{H}_{o'}|^2$ and $\mathbb{H}_{o'} = \{h \mid o' = M(h)\}$. □

Theorem 3.8. *$\mathscr{L}_{GE}$ is a observable hyperproperty.*
Proof: Let M be a program and q be a rational number such that $(M,q) \in \mathscr{L}_{GE}$. By the definition of the equivalence relation on outputs, there exists $T \in Obs$ such that $GE[U](T) = GE[U](M)$. By Lemma A.5, for any programs M', if $T \leq M'$, then $GE[U](M') > q$. Therefore, $\mathscr{L}_{GE}$ is a observable hyperproperty. □

Theorem 3.9. *Neither $\mathscr{U}_{SE}$ nor $\mathscr{L}_{SE}$ is a observable hyperproperty.*
Proof: Straightforward by Lemma A.3. □

Theorem 3.13. *Neither $\mathscr{L}_{ME}$ nor $\mathscr{L}_{GE}$ is a k-observable hyperproperty for any k.*
Proof:

- $\mathscr{L}_{ME}$

  For a contradiction, suppose $\mathscr{L}_{ME}$ is a k-observable hyperproperty. Let M be a program that has $k+1$ disjoint output traces. Then, We have $(M, \log(k+1)) \in \mathscr{L}_{ME}$. However, for any $T \in Obs$ such that $|T| \leq k$ and $T \leq M$, $ME[U](T) \leq \log k$. This leads to a contradiction.

- $\mathscr{L}_{GE}$

  For a contradiction, suppose $\mathscr{L}_{GE}$ is a k-observable hyperproperty. Let M be a program that has $k+1$ disjoint output traces. Then, We have $(M, \frac{k}{2}) \in \mathscr{L}_{GE}$. However, for any $T \in Obs$ such that $|T| \leq k$ and $T \leq M$, $GE[U](T) \leq \frac{k-1}{2}$. This leads to a contradiction.

□

Theorem 3.14. *Let q be a constant. Then, $\mathscr{L}_{ME}$ is a $\lfloor 2^q \rfloor + 1$-observable hyperproperty.*
Proof: Let M be a program such that $(M,q) \in \mathscr{L}_{ME}$. By Lemma 3.15, it must be the case that $|ran(M)| \geq \lfloor 2^q \rfloor + 1$ where $ran(M)$ is the range of M. Then, there exists $T \leq M$ such that $|T| \leq \lfloor 2^q \rfloor + 1$ and $ran(T) \geq \lfloor 2^q \rfloor + 1$. Then, by Lemma 3.15, it follows that for any program M' such that $T \leq M'$, $(M', q) \in \mathscr{L}_{ME}$. Therefore, $\mathscr{L}_{ME}$ is a $\lfloor 2^q \rfloor + 1$-observable property. □

Lemma A.6 *Let M be a program such that $GE[U](M) > q$, $\forall M'.M' \subsetneq M \Rightarrow GE[U](M') \leq q$, and $\forall M''.(GE[U](M'') > q \wedge (\forall M'.M' \subsetneq M'' \Rightarrow GE[U](M') \leq q)) \Rightarrow |M| \geq |M''|$. It must be the case that $|M| \leq \lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$.*

Proof: First, we prove M has exactly two outputs. Let n be an integer such that $n = |M|$. If M returns only one output, we have $GE[U](M) = 0$. Therefore, M must have more than 1 output, since $GE[U](M) > q$. We have for any o'

$$\begin{aligned} GE[U](M) &= \tfrac{n}{2} - \tfrac{1}{2n}(B + (n-i)^2) \\ &= i - \tfrac{1}{2n}(B + i^2) \end{aligned}$$

where $i = \sum_{o \in \mathbb{O} \setminus \{o'\}} |\mathbb{H}_o|$ and $B = \sum_{o \in \mathbb{O} \setminus \{o'\}} |\mathbb{H}_o|^2$. Because $GE[U](M) > q$, we have $i > q$. Then, we have

$$\begin{aligned} GE[U](M) > q \quad & \text{iff } i - \tfrac{B+i^2}{2n} > q \\ & \text{iff } n > \tfrac{B+i^2}{2(i-q)} \end{aligned}$$

By the definition of M, we have $\forall M'.M' \subsetneq M \Rightarrow GE[U](M') \leq q$. Let $\bar{M} = M \setminus \{(h', o')\}$ where $M(h') = o'$. Then, we have

$$\begin{aligned} GE[U](\bar{M}) \leq q \quad & \text{iff } i - \tfrac{B+i^2}{2(n-1)} \leq q \\ & \text{iff } n \leq \tfrac{B+i^2}{2(i-q)} + 1 \end{aligned}$$

Hence, we have

$$\frac{B+i^2}{2(i-q)} < n \leq \frac{B+i^2}{2(i-q)} + 1$$

$$
\begin{array}{l}
T(\psi) \equiv \\
\quad \mathsf{case}\ (H', \psi, \overrightarrow{H}) \\
\quad\quad \mathsf{when}\ (\mathsf{true}, \mathsf{true}, \_)\ \mathsf{then}\ \overrightarrow{O} := \overrightarrow{H}; O' := \mathsf{true}; O'' := \mathsf{true} \\
\quad\quad \mathsf{when}\ (\mathsf{true}, \mathsf{false}, \_)\ \mathsf{then}\ \overrightarrow{O} := \overrightarrow{\mathsf{true}}; O' := \mathsf{true}; O'' := \mathsf{false} \\
\quad\quad \mathsf{when}\ (\mathsf{false}, \_, \overrightarrow{\mathsf{true}})\ \mathsf{then}\ \overrightarrow{O} := \overrightarrow{\mathsf{true}}; O' := \mathsf{false}; O'' := \mathsf{false} \\
\quad\quad \mathsf{else} \\
\quad\quad\quad \mathsf{if}\ H_1 \\
\quad\quad\quad\quad \mathsf{then}\ \overrightarrow{O} := \overrightarrow{H}; O' := \mathsf{false}; O'' := \mathsf{true} \\
\quad\quad\quad\quad \mathsf{else}\ \ \overrightarrow{O} := \overrightarrow{\mathsf{true}}; O' := \mathsf{false}; O'' := \mathsf{false}
\end{array}
$$

where H', $\overrightarrow{H} = H_1, \ldots, H_n$, and O', O'', $\overrightarrow{O}$ are distinct.

Figure 3: The Loop-free Boolean Program for Theorem 3.17.

Because $B = \sum_{o \in \mathbb{O} \setminus \{o'\}} |\mathbb{H}_o|^2$ and $i = \sum_{o \in \mathbb{O} \setminus \{o'\}} |\mathbb{H}_o|$, the cargest n occurs when $B = i^2$. That is, when M has exactly two outputs. Next, we prove $|M| \leq \lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$. Recall that $i = \sum_{o \in \mathbb{O} \setminus \{o'\}} |\mathbb{H}_o|$. Let $j = n - i$. We have

$$
\begin{array}{rl}
GE[U](M) & = i - \frac{1}{2^n}(i^2 + i^2) \\
& = i - \frac{i^2}{n} \\
& = (n - j) - \frac{(n-j)^2}{n} \\
& = j - \frac{j^2}{n} \\
& > q
\end{array}
$$

This means that $j > q$. Recall that $\bar{M} = M \setminus \{(h', o')\}$ where $M(h') = o'$. Then, we have

$$
\begin{array}{ll}
GE[U](\bar{M}) \leq q & \text{iff } i - \frac{i^2}{n-1} \leq q \\
& \text{iff } n \leq \frac{i^2}{i-q} + 1
\end{array}
$$

Because n is an integer, we have $n \leq \lfloor \frac{i^2}{i-q} \rfloor + 1$ and $n \leq \lfloor \frac{j^2}{j-q} \rfloor + 1$. Let $f = \frac{i^2}{i-q} + 1 = \frac{j^2}{j-q} + 1$. By elementary real analysis, it can be shown that for integers i and j such that $i > q$ and $j > q$, f attains its maximum value when $i = \lfloor q \rfloor + 1$ or $j = \lfloor q \rfloor + 1$. Therefore, it follows that $|M| = n \leq \lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$. □

Theorem 3.16. *Let q be a constant. Then, $\mathscr{L}_{GE}$ is a $\lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$-observable hyperproperty.*
Proof: Let M be a program and q be a rational number such that $(M, q) \in \mathscr{L}_{GE}$. By Lemma A.6, there exists T such that $T \leq M$ and $|T| \leq \lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$. By Lemma A.5, for any M' such that $T \leq M'$, $GE[U](M') > q$. □

Theorem 3.17. *$\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are PP-hard for loop-free boolean programs.*
Proof:

- $\mathscr{L}_{ME}$
  We prove by reducing MAJSAT to $\mathscr{L}_{ME}$. Let ϕ be a boolean formula. Let q be the rational number such that

$$
\begin{array}{rl}
q & = ME[U](T(\psi)) \\
& = \log(2^{n-1} + 1 + 2^{n-1} - 1) \\
& = n - 1
\end{array}
$$

where n is the number of boolean variables in ϕ, and ψ is a boolean formulas such that $\#SAT(\psi) = 2^{n-1}$. We have

$$\begin{aligned} ME[U](T(\phi)) > q \quad &\Leftrightarrow ME[U](T(\phi)) \leq ME[U](T(\psi)) \\ &\Leftrightarrow \#SAT(\phi) \geq 2^{n-1}+1 \\ &\Leftrightarrow \#SAT(\phi) > 2^{n-1} \\ &\Leftrightarrow \phi \in \text{MAJSAT} \end{aligned}$$

by Lemma 3.15. Therefore, we can decide if $\phi \in$ MAJSAT by deciding if $ME[U](T(\phi)) \leq q$. Note that $T(\phi)$ and q can be constructed in time polynomial in the size of ϕ. Therefore, this is a reduction from MAJSAT to $\mathscr{L}_{ME}$.

- $\mathscr{L}_{GE}$

  We prove by reducing MAJSAT, which is a *PP*-complete problem, to $\mathscr{L}_{GE}$. Let ϕ be a boolean formula. Let q be the rational number such that

$$\begin{aligned} q &= GE(O := \psi \vee H) \\ &= \tfrac{2^{n+1}}{2} - \tfrac{1}{2^{n+2}}(|M^{-1}(\mathsf{true})|^2 + |M^{-1}(\mathsf{false})|^2) \\ &= 2^n - \tfrac{1}{2^{n+2}}((2^{n-1})^2 + (2^{n+1}-2^{n-1})^2) \end{aligned}$$

  where n is the number of boolean variables in ϕ, and ψ is a boolean formula such that $\#SAT(\psi) = 2^{n-1}$, and H be a boolean variable that does not appear in ψ and ϕ. We have

$$\begin{aligned} GE[U](O := \phi \vee H) > q \quad &\Leftrightarrow GE[U](O := \phi \vee H) > GE[U](O := \psi \vee H) \\ &\Leftrightarrow GE[U](O := \phi \vee H) > q \\ &\Leftrightarrow \#SAT(\phi) > \#SAT(\psi) \\ &\Leftrightarrow \#SAT(\phi) > 2^{n-1} \\ &\Leftrightarrow \phi \in \text{MAJSAT} \end{aligned}$$

  by Lemma A.5. Therefore, we can decide if $\phi \in$ MAJSAT by deciding if $GE[U](O := \phi \vee H) > q$. Note that $O := \phi \vee H$ and q can be constructed in time polynomial in the size of ϕ. Therefore, this is a reduction from MAJSAT to $\mathscr{L}_{GE}$.

□

Theorem 3.18. *Let q be a constant. Then, $\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are NP-complete for loop-free boolean programs.*

Proof:

- $\mathscr{L}_{ME}$

  $\mathscr{L}_{ME}$ is *NP*-hardness follows from the fact that $ME[U](O := \phi \wedge H) > 0$ iff $\phi \in$ SAT where H is a boolean variable that does not appear in ϕ.

  Next, we show that $\mathscr{L}_{ME} \in NP$. Recall that k-observable hyperproperties can be reduced to 1-observable hyperproperties via self composition. We can decide if the information flow of a given program M is greater than q by checking if the predicate of the assert statement fails for some inputs in the following program.

$$\begin{aligned} &M'(H_1, H_2, \ldots, H_n) \equiv \\ &\quad O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n); \\ &\quad \mathsf{assert}(\textstyle\bigvee_{i,j \in \{1,\ldots,n\}} (O_i = O_j \wedge i \neq j)) \end{aligned}$$

where $n = \lfloor 2^q \rfloor + 1$. That is, if for some inputs, the weakest precondition of M' does not hold, then $ME[U](M) > q$. Since weakest preconditions of loop-free boolean programs are boolean formulae over boolean variables representing inputs, we can reduce the problem of deciding if $ME[U](M) > q$ to SAT.

- $\mathscr{L}_{GE}$

  We have that $GE[U](O := \phi \wedge H) > 0$ iff $\phi \in$ SAT where H is a variable that does not appear in ϕ. It follows that $\mathscr{L}_{GE}$ is *NP*-hard.

  Next, we show $\mathscr{L}_{GE} \in NP$

$$
\begin{array}{l}
M'(H_1, H_2, \ldots, H_n) \equiv \\
\quad O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n); \\
\quad \mathsf{assert}(\sum_o |\{O_i \mid O_i = o\}|^2 < 2n(\frac{n}{2} + q))
\end{array}
$$

  where $n = \lfloor \frac{(\lfloor q \rfloor + 1)^2}{\lfloor q \rfloor + 1 - q} \rfloor + 1$.[9] That is, if for some inputs, the weakest precondition of M' does not hold, then $GE[U](M) > q$. Since weakest preconditions of loop-free boolean programs are boolean formulae over boolean variables representing inputs, we can reduce the problem of deciding if $GE[U](M) > q$ to SAT.

□

Theorem A.7 *Checking non-interference is PSPACE-complete for boolean programs.*

Proof: Straightforward from the fact that the reachability for boolean programs is PSPACE-complete. □

Theorem 3.19. *$\mathscr{L}_{ME}$ and $\mathscr{L}_{GE}$ are PSPACE-complete for boolean programs.*
Proof: *PSPACE*-hardness follows from Theorem 2.8 and Theorem A.7.

Next, we show that $\mathscr{L}_{ME} \in PSPACE$. To prove that $ME[U](M) > q$, it suffices to show that the number of outputs is greater than $\lfloor 2^q \rfloor + 1$. Let n be the number of inputs. The number of outputs can be enumerated in polynomial space, because the number of outputs is less than or equal to 2^n, and the problem of deciding if a program has a particular output can be solved in polynomial space. Therefore, we have $\mathscr{L}_{ME} \in PSPACE$.

Finally, we show that $\mathscr{L}_{GE} \in PSPACE$. By Lemma A.4, we have $\sum_o |\mathbb{H}_o|^2 < 2n(\frac{n}{2} + q)$ iff $GE[U](M) > q$. Because $\sum_o |\mathbb{H}_o|^2$ can be calculated in polynomial space, it follows the result. □

Theorem 4.2. *U_{ME} and U_{GE} are safety hyperproperties.*
Proof: Straightforward by Lemma 3.15 and Lemma A.5. □

Theorem 4.8. *Let q be a constant. Then, $\mathscr{U}_{ME}$ and $\mathscr{U}_{GE}$ are coNP-complete for loop-free boolean programs.*
Proof: By the similar manner to Theorem 3.19, we obtain the results. □

Theorem 4.9. *$\mathscr{U}_{ME}$ and $\mathscr{U}_{GE}$ are PSPACE-complete for boolean programs.*
Proof: This theorem is proven in the similar manner to Theorem 3.19. □

[9] Technically, the predicate of the assert statement is represented by a DNF of equalities over O_i. Therefore, the self-composed program can be constructed in time linear in the size of the original program.

$$\frac{ev(e,\sigma) = v}{(x := e, \sigma) \to \sigma[x \mapsto v]}$$

$$\frac{ev(e,\sigma) = \mathsf{true} \qquad (\mathsf{M}_0, \sigma) \to \sigma'}{(\texttt{if } e \texttt{ then } M_0 \texttt{ else } M_1, \sigma) \to \sigma'}$$

$$\frac{ev(e,\sigma) = \mathsf{false} \qquad (\mathsf{M}_1, \sigma) \to \sigma'}{(\texttt{if } e \texttt{ then } M_0 \texttt{ else } M_1, \sigma) \to \sigma'}$$

$$\frac{ev(e,\sigma) = \mathsf{false}}{(\texttt{while } e \texttt{ do } M, \sigma) \to \sigma}$$

$$\frac{ev(e,\sigma) = \mathsf{true} \qquad (\mathsf{M}, \sigma) \to \sigma'' \qquad (\texttt{while e do M}, \sigma'') \to \sigma'}{(\texttt{while } e \texttt{ do } M, \sigma) \to \sigma'}$$

$$\frac{}{(\texttt{skip}, \sigma) \to \sigma}$$

$$\frac{(M, \sigma) \to \sigma'' \qquad (M', \sigma'') \to \sigma'}{(M; M', \sigma) \to \sigma'}$$

Figure 4: The semantics of boolean programs