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 $M_1 \equiv \text{if } H = g \text{ then } O := 0 \text{ else } O := 1$ 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, 18, 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 $\mathcal{X}$ such that given a program $M$, $\mathcal{X}(M)$ is a non-negative real number. (Concretely, $\mathcal{X}$ is $SE[\mu]$ for the Shannon-entropy-based definition with the distribution $\mu$, $ME[\mu]$ for the min-entropy-based definition with the distribution $\mu$, and $GE[\mu]$ for the guessing-entropy-based definition with the distribution $\mu$.)

In a previous work [22, 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

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

To appear in EPTCS.
the hardness results to QIF verification problems was the notion of $k$-safety, which is an instance in a
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 hyper-
property (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 hyper-
property, 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.

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 uni-
miformly 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.

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 nev-
ethertheless 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 com-
plexity 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.

## 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], $H[\mu](X)$, which is the average of the information

1Technically, only non-empty observable hyperproperties are liveness hyperproperties.
2In fact, computing QIF under other input distributions can sometimes be reduced to this case [2]. See also Section 5.3
Table 1: A summary of hyperproperty classifications

<table>
<thead>
<tr>
<th></th>
<th>SE[U]</th>
<th>ME[U]</th>
<th>GE[U]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP</td>
<td>Liveness</td>
<td>Liveness</td>
<td>Liveness</td>
</tr>
<tr>
<td>UBP</td>
<td>Safety</td>
<td>Safety</td>
<td>Safety</td>
</tr>
<tr>
<td>LBP constant bound</td>
<td>Liveness $k$-observable</td>
<td>Liveness $k$-observable</td>
<td></td>
</tr>
<tr>
<td>UBP constant bound</td>
<td>Safety $k$-safety [31]</td>
<td>Safety $k$-safety [31]</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: A summary of computational complexities

<table>
<thead>
<tr>
<th></th>
<th>SE[U]</th>
<th>ME[U]</th>
<th>GE[U]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP for BP</td>
<td>PSPACE-hard</td>
<td>PSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>UBP for BP</td>
<td>PSPACE-hard</td>
<td>PSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>LBP for loop-free BP</td>
<td>PP-hard</td>
<td>PP-hard</td>
<td>PP-hard</td>
</tr>
<tr>
<td>LBP for loop-free BP, constant bound</td>
<td>Unknown</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td>UBP for loop-free BP, constant bound</td>
<td>Unknown</td>
<td>coNP-complete</td>
<td>coNP-complete</td>
</tr>
</tbody>
</table>

content, and intuitively, denotes the uncertainty of the random variable $X$. And, we review the notion of the conditional entropy, $\mathcal{H}[\mu](Y|Z)$, 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 $\mathcal{X}$ and $\mu$ be a probability distribution associated with $X$ (we write $\mu$ explicitly for clarity). The Shannon entropy of $X$ is defined as

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

Let $Y$ and $Z$ be random variables with sample space $\mathcal{Y}$ and $\mathcal{Z}$, respectively, and $\mu'$ be a probability distribution associated with $Y$ and $Z$. Then, the conditional entropy of $Y$ given $Z$ is defined as

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

where

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

$$\mu(Y = y|Z = z) = \frac{\mu(Y = y, Z = z)}{\mu(Z = z)}$$

(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 $\mu$ 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.)

$$\mu(O = o) = \sum_{h \in \mathcal{H}, M(h) = o} \mu(H = h)$$
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 \( \forall i \in \omega, o_i = \bot \lor o'_i = \bot \lor o_i = o'_i \) where \( o \) and \( o_i \) denotes the \( i \)th element of \( o \), and \( \bot \) is the special symbol denoting termination.\footnote{Here, we adopt the trace based QIF formalization of \cite{22}.}

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

\[
M(h) \text{ is equal to } o \iff \sigma_0; \sigma_1; \ldots; \sigma_i; \ldots \in M
\]

where \( \sigma_0(H) = h \) and \( \forall i \geq 1, \sigma_i(O) = o_i \) (\( o_i \) denotes the \( i \)th element of \( o \))

Here, \( \sigma \) 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 \( \sigma_0 \) and \( \sigma'_0 \) 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 \cite{13, 8, 21})** Let \( M \) be a program with a high security input \( H \), and a low security output trace \( O \). Let \( \mu \) be a distribution over \( H \). Then, the Shannon-entropy-based quantitative information flow is defined

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

Intuitively, \( \mathcal{H}[\mu](H) \) denotes the initial uncertainty and \( \mathcal{H}[\mu](H|O) \) 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 \footnote{Here, we adopt the trace based QIF formalization of \cite{22}.} 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.

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

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

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 \cite{28}.

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

\[
\mathcal{H}_\infty[\mu](X) = \log \frac{1}{\mathcal{Y}[\mu](X)}
\]

and the conditional min entropy of \( X \) given \( Y \) is defined

\[
\mathcal{H}_\infty[\mu](X|Y) = \log \frac{1}{\mathcal{Y}[\mu](X|Y)}
\]
where
\[
\mathcal{V}[\mu](X) = \max_{x \in \mathbb{X}} \mu(X = x)
\]
\[
\mathcal{V}[\mu](X|Y = y) = \max_{x \in \mathbb{X}} \mu(X = x|Y = y)
\]
\[
\mathcal{V}[\mu](X|Y) = \sum_{y \in \mathbb{Y}} \mu(Y = y) \mathcal{V}[\mu](X|Y = y)
\]

Intuitively, \(\mathcal{V}[\mu](X)\) 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 \(\mu\) be a distribution over \(H\). Then, the min-entropy-based quantitative information flow is defined
\[
ME[\mu](M) = H[\mu](H) - H[\mu](H|O)
\]

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,
\[
ME[U](M_1) = H[U](H) - H[U](H|O) = \log 4 - \log 2 = 1
\]
\[
ME[U](M_2) = H[U](H) - H[U](H|O) = \log 4 - \log 1 = 2
\]

Again, we have that \(ME[U](M_1) \leq ME[U](M_2)\) and \(ME[U](M_2) \leq ME[U](M_1)\), 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 \(\mu\) be an associated probability distribution. Then, the guessing entropy of \(X\) is defined
\[
\mathcal{G}[\mu](X) = \sum_{1 \leq i \leq m} i \times \mu(X = x_i)
\]

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

The conditional guessing entropy of \(X\) given \(Y\) is defined
\[
\mathcal{G}[\mu](X|Y) = \sum_{y \in \mathbb{Y}} \mu(Y = y) \sum_{1 \leq i \leq m} i \times \mu(X = x_i|Y = y)
\]

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

Intuitively, \(\mathcal{G}[\mu](X)\) 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 \(\mu\) be a distribution over \(H\). Then, the guessing-entropy-based quantitative information flow is defined
\[
GE[\mu](M) = \mathcal{G}[\mu](H) - \mathcal{G}[\mu](H|O)
\]

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.
\[
GE[U](M_1) = \mathcal{G}[U](H) - \mathcal{G}[U](H|O) = \frac{5}{2} - \frac{7}{4} = 0.75
\]
\[
GE[U](M_2) = \mathcal{G}[U](H) - \mathcal{G}[U](H|O) = \frac{5}{2} - 1 = 1.5
\]

Therefore, we again have that \(GE[U](M_1) \leq GE[U](M_2)\) and \(GE[U](M_2) \leq GE[U](M_1)\), 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 \( q \).

\[
U_{\text{SE}} = \{ (M, q) \mid \text{SE}[U](M) \leq q \}
\]

\[
U_{\text{ME}} = \{ (M, q) \mid \text{ME}[U](M) \leq q \}
\]

\[
U_{\text{GE}} = \{ (M, q) \mid \text{GE}[U](M) \leq q \}
\]

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 \( q \).

\[
L_{\text{SE}} = \{ (M, q) \mid \text{SE}[U](M) > q \}
\]

\[
L_{\text{ME}} = \{ (M, q) \mid \text{ME}[U](M) > q \}
\]

\[
L_{\text{GE}} = \{ (M, q) \mid \text{GE}[U](M) > q \}
\]

2.3 Non Interference

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

**Definition 2.7 (Non-interference [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 \( (M, 0) \in U_{\text{SE}} \). 2.) \( M \) is non-interfering iff \( (M, 0) \in U_{\text{ME}} \). 3.) \( M \) is non-interfering iff \( (M, 0) \in U_{\text{GE}} \).

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 \( P \subseteq \mathcal{P}(\Psi_{\text{inf}}) \) where \( \Psi_{\text{inf}} \) is the set of all infinite traces, and \( \mathcal{P}(X) \) 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,

\[
\begin{align*}
\text{Obs} & = \mathcal{P}^{\text{fin}}(\Psi_{\text{fin}}) \\
\text{Prop} & = \mathcal{P}(\Psi_{\text{inf}})
\end{align*}
\]

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

\[
S \leq T \iff \forall t \in S, \exists \forall t' \in T
\]

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 $S \in \text{Obs}$, there exists a set of traces $S' \in \text{Prop}$ 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.

Theorem 3.3 $L_{\text{SE}}, L_{\text{ME}},$ and $L_{\text{GE}}$ are liveness hyperproperties.\footnote{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.}

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.\footnote{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.}

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

Theorem 3.4 $U_{\text{SE}}$ is a liveness hyperproperty.\footnote{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.}

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 to sets of traces.

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 $T \in \text{Obs}$ such that $T \leq S$, and for any set of traces $S' \in \text{Prop}$, $T \leq S' \Rightarrow S' \in P$.

We call $T$ in the above definition an evidence.\footnote{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.}

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 $T \in \text{Obs}$ such that $T \leq M$ and $\forall M' \in \text{Prop}. T \leq M' \Rightarrow M' \in P$. For any set of traces $S \in \text{Obs}$, there exists $M' \in \text{Prop}$ 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. \Box

We note that the empty set is not a liveness hyperproperty but an observable hyperproperty.\footnote{We implicitly extend the notion of hyperproperties to classify hyperproperties that take programs and rational numbers. See [31].}

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

Theorem 3.7 $L_{\text{ME}}$ is an observable hyperproperty.

Theorem 3.8 $L_{\text{GE}}$ is an observable hyperproperty.
Theorem 3.7 follows from the fact that, if \((M, q) \in \mathcal{L}_{ME}\), then \(M\) contains an evidence of \(\mathcal{L}_{ME}\). 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 \(U_{SE}\) nor \(L_{SE}\) is an observable hyperproperty. We give the intuition of the proof for \(U_{SE}\). 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 \(L_{SE}\) 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 \(T \in \text{Obs}\) such that \(T \leq S, |T| \leq k\), and for any set of traces \(S' \in \text{Prop}\), \(T \leq S' \Rightarrow S' \in P\).

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.

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

where \(s[i]\) denotes the \(i\)th 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.

\[\{M \mid \exists h, h'. M(h) = 1 \land M(h') = 2\}\]

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, \(\parallel\) denotes a parallel composition.)

\[M'(H, H') \equiv O := M(H) \parallel O' := M(H') \parallel \text{assert}(\neg(O = 1 \land O' = 2))\]

Clearly, \(M\) satisfies the property iff the assertion failure is reachable in the above program, that is, iff the predicate \(O = 1 \land O' = 2\) 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 $\mathcal{L}_{\text{ME}}$ nor $\mathcal{L}_{\text{GE}}$ 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, $\mathcal{L}_{\text{ME}}$ 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, $(M,q) \in \mathcal{L}_{\text{ME}}$ iff there is an evidence in $M$ containing $\lfloor 2^q \rfloor + 1$ disjoint outputs.

Lemma 3.15  $\mathcal{L}_{\text{ME}} |_{\text{U}} (M) = \log \vert \{ o \mid \exists h. M(h) = o \} \vert$

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

Theorem 3.16  Let $q$ be a constant. Then, $\mathcal{L}_{\text{GE}}$ is a $\lfloor (\lfloor q \rfloor + 1)^2 \rfloor + 1$-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 $\mathcal{L}_{\text{ME}}$ and $\mathcal{L}_{\text{GE}}$ 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.

$$
M ::= x := \psi | M_0 ; M_1 | \text{if } \psi \text{ then } M_0 \text{ else } M_1 | \text{while } \psi \text{ do } M | \text{skip} \\
\phi, \psi ::= \text{true} | x | \phi \land \psi | \neg \phi
$$

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  $\mathcal{L}_{\text{ME}}$ and $\mathcal{L}_{\text{GE}}$ 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 $\phi$ over variables $\bar{x}$, if there are more than $2^{\vert \bar{x} \vert - 1}$ satisfying assignments to $\phi$.

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, $\mathcal{L}_{\text{ME}}$ and $\mathcal{L}_{\text{GE}}$ 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 $L_{ME}$ and $L_{GE}$ for a constant $q$ are in NP follows from the fact that $L_{ME}$ and $L_{GE}$ are $k$-observable hyperproperties for some $k$. We give the proof intuition for $L_{ME}$. 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.

$$M'(H_1, H_2, \ldots, H_n) \equiv$$

$$O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n);$$

$$\text{assert}(\bigvee_{i,j \in \{1, \ldots, n\}} (O_i = O_j \land i \neq j))$$

where $n = \lceil 2^q \rceil + 1$. Let $\phi$ 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 $\bigvee_{i,j \in \{1, \ldots, n\}} (O_i = O_j \land i \neq j)$. 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, deciding $ME[U](M) > q$ is reducible to SAT.

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

**Theorem 3.19** $L_{ME}$ and $L_{GE}$ are PSPACE-complete for boolean programs.

**Theorem 3.20** $L_{SE}$ 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 $T \in \text{Obs}$ such that $T \leq S$, and $\forall S' \in \text{Prop}. T \leq S' \Rightarrow S' \notin P$.

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 $T \in \text{Obs}$ such that $T \leq S$, $|T| \leq k$, and $\forall S' \in \text{Prop}. T \leq S' \Rightarrow S' \notin P$.

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].

---

8For 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 $U_{ME}$ and $U_{GE}$ are $k$-safety hyperproperties when the bound $q$ is fixed to a constant.

Theorem 4.5 ([31]) Let $q$ be a constant. $U_{ME}$ is a $\left\lfloor \frac{\left\lceil q \right\rceil + 1}{q} \right\rfloor + 1$-safety property.

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

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

$$M'(H_1, H_2, \ldots, H_n) \equiv$$
$$O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n);$$
$$\text{assert} \left( \bigvee_{i,j \in \{1, \ldots, n\}} (O_i = O_j \land i \neq j) \right)$$

where $n = \left\lfloor 2^q \right\rfloor + 1$. 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 $\bigvee_{i,j \in \{1, \ldots, n\}} (O_i = O_j \land i \neq j)$ 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, $U_{ME}$ is in coNP.

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

Theorem 4.9 $U_{ME}$ and $U_{GE}$ are PSPACE-complete for boolean programs.

Theorem 4.10 $U_{SE}$ 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 $\sigma$). 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 $T \in \text{Obs}$, there exists $T' \in \text{Prop}$ 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., $\max_\mu \text{SE}[\mu]$). 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, $\max_\mu \text{SE}[\mu] = \text{ME}[U]$. Therefore, we obtain the same hyperproperty classifications and complexity results for channel capacity as $\text{ME}[U]$.

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., $\max_\mu \text{ME}[\mu]$ and $\max_\mu \text{GE}[\mu]$). It has been shown that $\max_\mu \text{ME}[\mu] = \text{ME}[U]$ [5, 19] and $\max_\mu \text{GE}[\mu] = \text{GE}[U]$ [33], that is, they attain their maximums when the distributions are uniform. Therefore, they have the same hyperproperty classifications and complexities as $\text{ME}[U]$ and $\text{GE}[U]$, 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 $\text{PSPACE}$-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


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.
Quantitative Information Flow as Safety and Liveness 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.

- \(\mathcal{LE}(1)\): \(\mathcal{L}_{ME}\) under the constant bound 1.
- \(\mathcal{GE}(1)\): \(\mathcal{L}_{GE}\) under the constant bound 1.
- \(\mathcal{ME}(1)\): \(\mathcal{L}_{ME}\) under the constant bound 1.
- \(\mathcal{GE}(1)\): \(\mathcal{L}_{GE}\) under the constant bound 1.
- **NI**: the set of all non-interfering programs.

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

### A.1 Omitted Proof

**Definition A.1**

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

Intuitively, \(\text{In}(\mu, X, x)\) is the order of \(x\) defined in terms of \(\mu\).

**Lemma A.2**

\[\mathcal{F}[\mu](X) = \Sigma_{1 \leq i \leq |X|} i \mu(x_i) = \Sigma_{x \in X} \text{In}(\mu, X, x) \mu(x)\]

**Proof:** Trivial. \(\square\)

**Theorem 2.8.**

1.) \(M\) is non-interfering iff \((M, 0) \in \mathcal{SE}\). 2.) \(M\) is non-interfering iff \((M, 0) \in \mathcal{ME}\). 3.) \(M\) is non-interfering iff \((M, 0) \in \mathcal{GE}\).

**Proof:** Let \(\emptyset = \{M(h, \ell) \mid h \in H \wedge \ell \in L\}\).

- **SE**
  
  (See [7].)

- **ME**
  
  Suppose \(M\) is non-interfering. By definition, it suffices to show that

  \[\mathcal{F}[U](H) = \mathcal{F}[U](H|O)\]

  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

\[
\sum_o U(o) \max_h U(h | o) = \max_h \frac{U(h, o')}{U(h)} = \max_h U(h)
\]

\[
\Rightarrow
\]

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

\[
\sum_o U(o) \max_h U(h | o) = \sum_{o \in \emptyset \setminus \{o_1, o_2\}} \max_h U(h, o) + \max_h U(h, o_1) + \max_h U(h, o_2)
\]

And,

\[
\max_h U(h) = \max U(h, o_1)
\]

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

\bullet GE

\[
\Rightarrow
\]

Suppose $M$ is non-interfering. By definition,

\[
GE[U](M) = \sum_h \ln(\lambda h'. U(h'), \mathbb{H}, h) U(h) - \sum_{o} \sum_h \ln(\lambda h'. U(h', o), \mathbb{H}, h) U(h, o)
\]

\[
= \sum_h \ln(\lambda h'. U(h'), \mathbb{H}, h) U(h) - \sum_h \ln(\lambda h'. U(h', \mathbb{H}), \mathbb{H}, h) U(h)
\]

\[
= 0
\]

since for all $h_\chi$ and $o_\chi$ such that $U(h_\chi, o_\chi) > 0$, for any $h'_\chi$ and $o' \in \emptyset \setminus \{o_\chi\}$, $U(h'_\chi, o'_\chi) = 0$.

\[
\Rightarrow
\]

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,

\[
GE[U](M) = \sum_h \ln(\lambda h'. U(h'), \mathbb{H}, h) U(h) - \sum_{o} \sum_h \ln(\lambda h'. U(h', o), \mathbb{H}, h) U(h, o)
\]

\[
= A + \sum_h \ln(\lambda h'. U(h'), \mathbb{H}, h) U(h) - B - \sum_{o \in \emptyset} \sum_h \ln(\lambda h'. U(h', o), \mathbb{H}, h) U(h, o)
\]

where

\[
A = \sum_h \ln(\lambda h'. U(h'), \mathbb{H}, h) U(h)
\]

\[
B = \sum_{o \in \emptyset} \sum_h \ln(\lambda h'. U(h', o), \mathbb{H}, h) U(h, o)
\]

Trivially, we have $A \geq B$ and

\[
\sum_h \ln(\lambda h'. U(h'), \mathbb{H}, h) U(h) > \sum_{o} \sum_h \ln(\lambda h'. U(h', o), \mathbb{H}, h) U(h, o)
\]

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 \(T \in \text{Obs}\), 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
- \(\mathbb{H} = \text{dom}(M)\),
- \(\mathbb{H}' \cap \mathbb{H} = \emptyset\),
- 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 \(h' \in \mathbb{H}'\), for any \(h \in \mathbb{H}\), \(o_{h'} \neq M(h)\).

It follows that
\[
SE[U](M') \leq SE[U](M_1)
\]
And, we have
\[
SE[U](M_1) = \frac{|\mathbb{H}|}{|\mathbb{H}|^2 + |\mathbb{H}'|} \log(|\mathbb{H}| + |\mathbb{H}'|) + \frac{|\mathbb{H}'|}{|\mathbb{H}| + |\mathbb{H}'|} \log \frac{|\mathbb{H}| + |\mathbb{H}'|}{|\mathbb{H}'|}
\]
\[
\rightarrow 0 \quad (\|\mathbb{H}'\| \rightarrow \infty)
\]
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
- \(\mathbb{H} = \text{dom}(M)\),
- \(\mathbb{H}' \cap \mathbb{H} = \emptyset\),
- 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
\[
SE[U](M_1) = \frac{|\mathbb{H}'|}{|\mathbb{H}| + |\mathbb{H}'|} \log(|\mathbb{H}| + |\mathbb{H}'|) + \frac{|\mathbb{H}|}{|\mathbb{H}|^2 + |\mathbb{H}'|} \log \frac{|\mathbb{H}| + |\mathbb{H}'|}{|\mathbb{H}'|}
\]
\[
\rightarrow \infty \quad (\|\mathbb{H}'\| \rightarrow \infty)
\]
Therefore, there exists \(\mathbb{H}'\) such that \(SE[U](M'') > q\).

\(\square\)
Theorem 3.4. \( \mathcal{L}_{SE}, \mathcal{L}_{ME}, \) and \( \mathcal{L}_{GE} \) are liveness hyperproperties.

**Proof:** The fact that \( \mathcal{L}_{SE} \) is a liveness hyperproperty follows from Lemma A.3. The results for \( \mathcal{L}_{ME} \) and \( \mathcal{L}_{GE} \) follow from Theorem 3.7, Theorem 3.8, and Theorem 3.6. \( \square \)

Theorem 3.7. \( \mathcal{L}_{ME} \) is a observable hyperproperty.

**Proof:** Let \( M \) be a program and \( q \) be a rational number such that \((M, q) \in \mathcal{L}_{ME} \). By the definition of the equivalence relation on outputs, there exists \( T \in \text{Obs} \) 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, \( \mathcal{L}_{ME} \) is a observable hyperproperty. \( \square \)

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

**Proof:** By the definition, we have

\[
GE[U](M) = \mathcal{G}[U](H) - \mathcal{G}[U](H|O)
= \sum_i \mathcal{I}_i[U](H, h)U(h) - \sum_o U(o) \sum_i \mathcal{I}_i[U](h', U(h'|o), h)U(h'|o)
= \frac{1}{2} n(n + 1) - \sum_o |H_o| \cdot \frac{1}{2} |H_o| |H_o| + 1
= \frac{n}{2} + \frac{1}{2} - \frac{1}{2n} \sum_o |H_o|^2
= \frac{n}{2} - \frac{1}{2} \sum_o |H_o|^2
\]

\( \square \)

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

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

By Lemma A.4, we have

\[
GE[U](M') - GE[U](M)
= \frac{n + 1}{2} - \frac{1}{2n + 1} (B + (|H_o| + 1)^2) - \frac{n}{2} + \frac{1}{2n} (B + |H_o|^2)
= \frac{1}{2} + \frac{1}{2n + 1} (-n(B + (|H_o| + 1)^2) + (n + 1)(B + |H_o|^2))
= \frac{1}{2n + 1} (n(n + 1) - n(|H_o| + 1)^2 + B + (n + 1)(|H_o|^2))
= \frac{1}{2n + 1} (n^2 - 2n|H_o| + |H_o|^2 + B)
\geq 0
\]

where \( B = \sum_{o \in O \setminus \{o\}} |H_o'|^2 \) and \( H_o' = \{ h \mid o' = M(h) \} \). \( \square \)

Theorem 3.8. \( \mathcal{L}_{GE} \) is an observable hyperproperty.

**Proof:** Let \( M \) be a program and \( q \) be a rational number such that \((M, q) \in \mathcal{L}_{GE} \). By the definition of the equivalence relation on outputs, there exists \( T \in \text{Obs} \) 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, \( \mathcal{L}_{GE} \) is a observable hyperproperty. \( \square \)
**Theorem 3.9** Neither $\mathcal{H}_{SE}$ nor $\mathcal{L}_{SE}$ is a observable hyperproperty.

**Proof:** Straightforward by Lemma A.3 □

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

**Proof:**

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

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

□

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

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

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

**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 $\text{GE}[U](M) = 0$. Therefore, $M$ must have more than 1 output, since $\text{GE}[U](M) > q$. We have for any $o'$

$$\text{GE}[U](M) = \frac{n}{2} - \frac{1}{2n} \left( B + (n - i)^2 \right) = i - \frac{1}{2n} \left( B + i^2 \right)$$

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

$$\text{GE}[U](M) > q \quad \text{iff} \quad i - \frac{B + i^2}{2n} > q \quad \text{iff} \quad n > \frac{B + i^2}{2(i-q)}$$

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

$$\text{GE}[U](\tilde{M}) \leq q \quad \text{iff} \quad i - \frac{B + i^2}{2(n-1)} \leq q$$

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

Hence, we have

$$\frac{B + i^2}{2(i-q)} < n \leq \frac{B + i^2}{2(i-q)} + 1$$
Theorem 3.16. Let \( q \) be a constant. Then, \( \mathcal{L}_{GE} \) is a \( \lceil \frac{(q+1)^2}{|q|+1-q} \rceil + 1 \)-observable hyperproperty.

Proof: Let \( M \) be a program and \( q \) be a rational number such that \( (M, q) \in \mathcal{L}_{GE} \). By Lemma A.6, there exists \( T \) such that \( T \leq M \) and \( |T| \leq \lceil \frac{(|q|+1)^2}{|q|+1-q} \rceil + 1 \). By Lemma A.5, for any \( M' \) such that \( T \leq M' \), \( \mathcal{L}_{ME} \) and \( \mathcal{L}_{GE} \) are PP-hard for loop-free boolean programs.

Proof:

- \( \mathcal{L}_{ME} \)

We prove by reducing MAJSAT to \( \mathcal{L}_{ME} \). Let \( \phi \) be a boolean formula. Let \( q \) be the rational number such that

\[
q = \text{ME}[U](T(\psi)) = \log(2^{n-1} + 1 + 2^{n-1} - 1) = n - 1
\]
where \( n \) is the number of boolean variables in \( \phi \), and \( \psi \) is a boolean formulas such that \( \#\text{SAT}(\psi) = 2^{n-1} \). We have

\[
\text{ME}[U](T(\phi)) > q \iff \text{ME}[U](T(\phi)) \leq \text{ME}[U](T(\phi)) \\
\iff \#\text{SAT}(\phi) \geq 2^{n-1} + 1 \\
\iff \#\text{SAT}(\phi) > 2^{n-1} \\
\iff \phi \in \text{MAJSAT}
\]

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

- \( \mathcal{L}_{\text{GE}} \)

We prove by reducing MAJSAT, which is a \( \text{PP} \)-complete problem, to \( \mathcal{L}_{\text{GE}} \). Let \( \phi \) be a boolean formula. Let \( q \) be the rational number such that

\[
q = \text{GE}(O := \psi \lor H) \\
= 2^{n+1} - \frac{1}{2^{n+2}}(|M^{-1}(\text{true})|^2 + |M^{-1}(\text{false})|^2) \\
= 2^n - \frac{1}{2^{n+2}}((2^{n-1})^2 + (2^{n+1} - 2^{n-1})^2)
\]

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

\[
\text{GE}[U](O := \phi \lor H) > q \iff \text{GE}[U](O := \phi \lor H) > \text{GE}[U](O := \psi \lor H) \\
\iff \text{GE}[U](O := \phi \lor H) > q \\
\iff \#\text{SAT}(\phi) > \#\text{SAT}(\psi) \\
\iff \#\text{SAT}(\phi) > 2^{n-1} \\
\iff \phi \in \text{MAJSAT}
\]

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

\[\square\]

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

**Proof:**

- \( \mathcal{L}_{\text{ME}} \)

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

Next, we show that \( \mathcal{L}_{\text{ME}} \in \text{NP} \). 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.

\[
M'(H_1, H_2, \ldots, H_n) \equiv \\
O_1 := M(H_1); O_2 := M(H_2); \ldots; O_n := M(H_n); \\
assert(\forall i, j \in \{1, \ldots, n\} (O_i = O_j \land i \neq j))
\]
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.

- \( \mathcal{L}_{GE} \)

We have that \( GE[U](O := \phi \land H) > 0 \) iff \( \phi \in \text{SAT} \) where \( H \) is a variable that does not appear in \( \phi \).

It follows that \( \mathcal{L}_{GE} \in \text{NP} \).

Next, we show \( \mathcal{L}_{ME} \in \text{PSPACE} \). 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 \( \mathcal{L}_{ME} \in \text{PSPACE} \).

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

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 \( \mathcal{L}_{ME} \) and \( \mathcal{L}_{GE} \) are PSPACE-complete for boolean programs.

Proof: PSPACE-hardness follows from Theorem 2.8 and Theorem A.7.

Next, we show that \( \mathcal{L}_{ME} \in \text{PSPACE} \). 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 \( \mathcal{L}_{ME} \in \text{PSPACE} \).

Finally, we show \( \mathcal{L}_{GE} \in \text{PSPACE} \). By Lemma A.4, we have \( \sum_o |H_o|^2 < 2n(\frac{q}{2} + q) \) iff \( GE[U](M) > q \). Because \( \sum_o |H_o|^2 \) 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, \( \mathcal{U}_{ME} \) and \( \mathcal{U}_{GE} \) are coNP-complete for loop-free boolean programs.

Proof: By the similar manner to Theorem 3.19 we obtain the results.

Theorem 4.9 \( \mathcal{U}_{ME} \) and \( \mathcal{U}_{GE} \) are PSPACE-complete for boolean programs.

Proof: This theorem is proven in the similar manner to Theorem 3.19.

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.
$ev(e, \sigma) = v$

$(x := e, \sigma) \rightarrow \sigma[x \mapsto v]$

$ev(e, \sigma) = true \quad (M_0, \sigma) \rightarrow \sigma'$

$(if \ e \ then \ M_0 \ else \ M_1, \sigma) \rightarrow \sigma'$

$ev(e, \sigma) = false \quad (M_1, \sigma) \rightarrow \sigma'$

$(if \ e \ then \ M_0 \ else \ M_1, \sigma) \rightarrow \sigma'$

$ev(e, \sigma) = false \quad (while \ e \ do \ M, \sigma) \rightarrow \sigma$

$ev(e, \sigma) = true \quad (M, \sigma) \rightarrow \sigma'' \quad (while \ e \ do \ M, \sigma'') \rightarrow \sigma'$

$(while \ e \ do \ M, \sigma) \rightarrow \sigma'$

$(skip, \sigma) \rightarrow \sigma$

$(M, \sigma) \rightarrow \sigma'' \quad (M', \sigma'') \rightarrow \sigma'

$(M; M', \sigma) \rightarrow \sigma'$

Figure 4: The semantics of boolean programs