The Complexity of Quantitative Information Flow Problems

Pavol Černý, Krishnendu Chatterjee, Thomas A. Henzinger
IST Austria
{*cernyp,krishnendu.chatterjee,tah*}*@ist.ac.at*

***Abstract*—In this paper, we investigate the computational complexity of quantitative information flow (QIF) problems. Information-theoretic quantitative relaxations of noninterference (based on Shannon entropy) have been introduced to enable more fine-grained reasoning about programs in situations where limited information flow is acceptable. The *QIF bounding problem* asks whether the information flow in a given program is bounded by a constant d. Our first result is that the QIF bounding problem is PSPACE-complete. The *QIF memoryless synthesis problem* asks whether it is possible to resolve nondeterministic choices in a given partial program in such a way that in the resulting deterministic program, the quantitative information flow is bounded by a given constant d. Our second result is that the QIF memoryless synthesis problem is also PSPACE-complete. The QIF memoryless synthesis problem generalizes to *QIF general synthesis problem* which does not impose the memoryless requirement (that is, by allowing the synthesized program to have more variables then the original partial program). Our third result is that the QIF general synthesis problem is EXPTIME-hard.**

***Keywords*-quantitative information flow, verification, synthesis, computational complexity**

I. Introduction

Security and confidentiality are growing concerns in software development [1], [2]. The problem of ensuring the confidentiality of data being processed by computing systems is long-standing (see [3] for an early approach), yet increasingly important, as systems are parts of larger networks. The nodes on the network need to interact, often by communicating sensitive data over the network and thus giving an opportunity to active or passive (eavesdropping) attackers to gain knowledge about secret data.

Noninterference is a security property often used to ensure confidentiality. Informally, it can be described as follows: if two input states share the same values of low-security variables (the variables visible to the observer), then the behaviors of the program executed from these states are indistinguishable by the observer. See [4] for a survey of the research on noninterference, and [5] for a Java-based programming language with a type system that supports information flow control based on noninterference. It is well-known that the noninterference requirement needs to be relaxed in various contexts. See [6] for a survey of methods for defining such relaxations via declassification.

A different systematic way of relaxing noninterference is to make the requirement quantitative. This allows reasoning about information flow at a finer granularity, and distinguishing between two programs when both exhibit interference, but one is intuitively significantly more secure than the other. Quantitative versions of noninterference have been formalized using the information-theoretic notions of Shannon entropy and mutual information ([7], [8], [9], [10], [11], [12]).

A classical example of a program where quantitative reasoning about information flow is needed is the following password-checking example:

`Program` P_1`: if` H`=`inp `then` O`:=0 else` O`:=1`

In this program, H is the high-security input representing a stored password, inp is a low-security input (possibly provided by the attacker), and O is a low-security output (visible to the attacker). It is clear that there is an information flow from high-security variables (high variables for short) to low-security variables (low variables), since by executing P_1, the attacker learns whether H is equal to inp. However, this information leak might be acceptable, whereas the information leak in the program P_2 below might not be.

`Program` P_2`:` O `:=` H

Verification. Let $SE[\mu](P)$ denote the Shannon-entropy based quantitative information flow from high to low variables of the program P. It is defined, roughly, as the amount by which the entropy of high variables decreases for the attacker when she provides inputs and observes the execution of the program. The parameter μ is the probability distribution over the high and low inputs (we also model the case where the low inputs are provided by the attacker).

Given a finite-state program P, a probability distribution μ, and a rational number d, the *quantitative information flow (QIF) bounding problem* is to decide whether $SE[\mu](P) \geq d$. In other words, we ask whether the entropy on the high variables (from the point of view of the attacker) decreases by at least d (which might make the program unacceptable in certain application contexts). This problem has been defined and studied in [10], [11]. The results in [11] show that for the special case of loop-free boolean programs, the problem is coNP-complete, and the problem was left open for general boolean programs. The complexity is given in terms of the size of the program text. We solve the open problem of [11] by presenting matching lower and upper bound for the general problem. We show that the QIF bounding problem is PSPACE-complete for general finite-state programs, with

the complexity in terms of the explicit-state version of the program (that is, the program is given as a transition relation on its state space). Thus we settle the complexity of one of the fundamental questions in quantitative information flow. All our complexity results are for the explicit-state version of the program, and since we consider programs that may have loops, there is an exponential translation required from the implicit-state version of the program (and the exponential translation is unavoidable in the worst case [13], [14]).

Synthesis. A very promising approach to the development of correct programs is *partial program synthesis*. The goal here is to allow the programmer to specify a part of her intent declaratively, by saying what needs to be done or what conditions need to be maintained. The synthesizer then constructs a program that satisfies the specification (see for example [15], [16], [17]). A *partial program* is a finite-state concurrent program which includes non-deterministic choices which the synthesizer has to resolve. A program is *allowed* by a partial program if it can be obtained by resolving the nondeterministic choices. We study the partial program synthesis problem in the context of quantitative information flow.

Let us consider the following partial program:

```
Partial program M:
n  = choice(1..10);
for i:= 1 to n do {
  read(inp);
  if H=inp then
     O:=0
  else { O:=1; break; }
}
```

It gives the user n possibilities to enter/guess the password. The programmer left the choice of n unspecified. The task of the synthesizer is to synthesize a program, where an attacker cannot gain too much information, i.e. a program P allowed by M, where $SE[\mu](P) < d$, for a given constant d. In this instance, it might be equally easy to specify the desired number of iterations as it is to specify the desired level of uncertainty of the attacker about H. However, if the queries the attacker can use are even a little more complex (for example if the attacker could test whether H is less than the current guess), it might be much easier to specify a bound on the uncertainty of the attacker.

Given a finite state partial program M, a probability distribution μ, and a rational number d, the *QIF memoryless synthesis problem* is to decide whether for every program P such that P is allowed by M, we have $SE[\mu](P) \geq d$. In other words, the question asks whether for every memoryless refinement (choosing one transition for each nondeterministic choice in M), the answer to the QIF bounding problem is yes. We prove that the QIF memoryless synthesis problem is also PSPACE-complete. The QIF memoryless synthesis problem generalizes to *QIF general synthesis problem* which does not impose the memoryless restriction. More specifically, the synthesized program can add variables to the partial program (and store information in these variables). This allows the synthesizer to produce more general solutions to the problem. Our third result is that the QIF general synthesis problem is EXPTIME-hard and can be solved in 2EXPTIME.

	Verification	Memoryless Synth.	Gen. Synth.
QIF	PSPACE-complete	PSPACE-complete	EXPTIME-hard in 2EXPTIME

Table I
MAIN COMPLEXITY RESULTS

Technical contribution. To obtain the complexity results we present automata models of partial-information deterministic and nondeterministic systems (refer them as PIDS and PINS, respectively) and establish the equivalence of PIDS and PINS with programs and partial programs, respectively. We present our lower and upper bound proofs for the automata models and from them derive the results for the information flow problems for programs. Our main technical contributions are as follows:

1) *Lower bound proof.* We reduce the membership problem for polynomial-space *alternating* Turing machines to the QIF general synthesis problem for PINS, and the membership problem for polynomial-space *deterministic* Turing machines to the QIF bounding problem for PIDS. Our lower bound proofs based on reductions of Turing machines are completely different from the lower bound proofs of [10], [11] that used validity and counting satisfiable instances of SAT formulas.
2) *Upper bound proof.* The PSPACE upper bound proofs are obtained by considering the computation tree unrolling of a deterministic system, and on-the-fly construction of a witness to show that the answer to the QIF bounding problem is yes. The upper bound for QIF memoryless synthesis problem uses the result of QIF bounding problem along with a polynomial guess to obtain the desired PSPACE upper bound. The 2EXPTIME upper bound follows from a double subset construction.

Summary of contributions. To summarize, the main contributions of this work are as follows:

1) We settle the complexity of the QIF bounding problem for general programs by providing matching upper and lower bounds; we show the problem is PSPACE-complete. Our result provides the precise computational complexity of one of the fundamental problems in quantitative information flow.
2) We study the QIF partial program synthesis problem and show that the memoryless synthesis problem is PSPACE-complete and the general synthesis problem

is EXPTIME-hard and can be solved in 2EXPTIME.

The main complexity results of the paper are summarized in Table I.

II. Partial Programs and Quantitative Information Flow Problems

In this section we first introduce partial programs, then notions of noninterference, and quantitative information flow, and finally the decision problems related to partial programs and quantitative information flow.

A. Partial Programs

In this section we define partial programs, programs, and their syntax and semantics.

Syntax. We present the syntax for partial programs. We start with the definition of guards and operations.

Guards and operations. Let H, L, and I be finite sets of variables (representing high security, low security, and input variables, respectively) ranging over finite domains. Informally, the high variables contain the secret (high-security) input; the low variables are visible publicly; the input variables are for input provided by the user (attacker). A *term* t is either a variable in H, L, or I, or is defined by $t_1 \ op \ t_2$, where t_1 and t_2 are terms and op is an operator. Formulas are defined by the following grammar, where t_1 and t_2 are terms and rel is a relational operator: $e := t_1 \ rel \ t_2 \mid e_1 \wedge e_2 \mid \neg e$. *Guards* are formulae over H, L, and I. *Operations* are simultaneous assignments to variables in L, where each variable is assigned a term over H, L, and I.

Partial programs and program. A *partial program* is a tuple $\langle Q, H, L, I, \delta, q_0 \rangle$, where:

- Q is a finite set of control locations, q_0 is an initial location;
- H, L and I are finite sets of high security, low security, and input variables respectively, ranging over finite domains;
- δ is a set of tuples of the form (q, g, a, q'), where q and q' are locations from Q, and g and a are *guards* and *operations* over variables in H, L and G.

The set of locations $Sk(C)$ of a partial program $C = \langle Q, H, L, I, \delta, q_0 \rangle$ is the subset of Q containing exactly locations where δ is non-deterministic, i.e., locations q where for a valuation of variables in H, L, and I, there is more than one transition whose guard evaluates to true.

A *program* is a partial program C, where the set $Sk(C)$ is empty. A program P is *allowed* by a partial program M if it can be obtained by removing the outgoing transitions from locations $Sk(M)$ of all the threads of M, so that at most one transition from every location is enabled for a given valuation of variables from H, L and I. A program is thus deterministic.

Note that programs and partial programs are defined using their control-flow graph — there is thus no restriction on the control-flow structures.

Semantics. We now present the semantics of partial programs in terms of transition systems.

Transition systems. A *transition system* is a tuple $\langle S, \Delta, S_0 \rangle$ where S is a finite set of states, $\Delta \subseteq S \times S$ is a set of transitions, and S_0 is the set of initial states.

The semantics of a partial program M is given in terms of a transition system, and we denote the transition system of a partial program M as $\mathsf{Tr}(M)$. The transition system $\mathsf{Tr}(M)$ is defined as follows:

- *State space.* A state s in the set of states S of $\mathsf{Tr}(M)$ contains a location Q a valuation of all variables in H, L, and I, and a boolean value $turn$. A state in S is initial, if the location it contains is initial.
- *Transitions.* The transition function Δ defines interleaving semantics for the partial program. There are two types of transitions: program transitions, which model one step of the program, and environment transitions, that model input from the environment via variables in I. Let (q, g, a, q') be a transition of M. There is a program transition from a state s to a state s' if (i) the value of $turn$ is false (ii) the location of M in s is q and the location of M in s' is q', (iii) the guard g evaluates to true in s, and (iv) the valuation of all variables of M in s', and is obtained from the valuation of variables in s by performing the operation a. There is an environment transition from state s to state s' in $\mathsf{Tr}(M)$ if and only if (i) the value of $turn$ is true and (ii) the valuations of variables in s and s' differ only in input variables.

B. Noninterference

In this subsection we recall the notion of noninterference.

Runs and traces. Let $\mathsf{Tr}(M) = \langle S, \Delta, S_0 \rangle$ be a transition system obtained from a program $M = \langle Q, H, L, I, \delta, q_0 \rangle$. A *run* r is a sequence of states $s_0 s_1 s_2 \ldots$ of $\mathsf{Tr}(M)$ such that for all $i \geq 0$ we have $(s_i, s_{i+1}) \in \Delta$. Given a run $s_0 s_1 s_2 \ldots$, a *trace* σ is a sequence of valuations of low and input variables of M defined as follows: $l_0 i_1 l_1 i_2 l_2 \ldots$, where l_0 is a valuation of low variables given by s_0, i_1 and l_1 are valuations of input and low variables given by s_1, and i_2 and l_2 are valuations of input and low variables given by s_2. If $\mathbb{H}$ is the set of valuations of high variables, $\mathbb{L}$ is the set of valuations of low variables, and $\mathbb{I}$ is the set of valuations of input variables, then a trace is a sequence in $(\mathbb{L} \times \mathbb{I})^* \times \mathbb{L}$.

Note that the program is deterministic, so given the initial valuation l of low variables, the initial valuation h of high variables, and the sequence $\kappa = i_1 i_2 \ldots$ of valuations of input variables, a run of the program, as well as the corresponding trace σ, are determined. We denote the obtained trace by $P(h, l, \kappa)$.

Noninterference. A program P is *noninterferent*, if for all valuations h and h' of high variables, and all valuations l of low variables, and all sequences κ of valuations of input variables, we have that $P(h, l, \kappa) = P(h', l, \kappa)$.

Let P be a program. Let $\mathbb{H}$ be the set of valuations of high variables, let $\mathbb{L}$ be the set of valuations of low variables, and let $\mathbb{I}$ be the set of valuations of input variables. An *observation-based strategy* (for choosing inputs) is a function that given a sequence in $(\mathbb{L} \times \mathbb{I})^* \times \mathbb{L}$ gives an input in $\mathbb{I}$.

C. Quantitative Information Flow

In this subsection we recall the notion of Shannon entropy and mutual information, and then present the quantitative information flow problem.

Shannon entropy and mutual information. We follow the presentation of [9], [10]. We refer the reader to these works for more details and intuition on the information-theoretic notions and the definition of quantitative information flow. Let X be a random variable with sample space $\mathbb{X}$ and μ be a probability distribution associated with X. The *Shannon entropy* of X is defined as

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

with the logarithm in base 2.

Let X and Y be random variables with sample spaces $\mathbb{X}$ and $\mathbb{Y}$ and let μ be a probability distribution associated with X and Y. The *conditional entropy* of X given Y is defined by:

$$\mathcal{H}[\mu](X|Y) = \sum_{y \in \mathbb{Y}} \mu(Y = y) \cdot \mathcal{H}[\mu](X|Y = y),$$

where

$$\mathcal{H}[\mu](X|)Y = y = \sum_{x \in \mathbb{X}} \mu(X = x|Y = y) \cdot \log \frac{1}{\mu(X = x|Y = y)}$$

and

$$\mu(X = x|Y = y) = \frac{\mu(X = x, Y = y)}{\mu(Y = y)}$$

Let X, Y and Z be random variable with sample spaces $\mathbb{X}$, $\mathbb{Y}$, and $\mathbb{Z}$ and let μ be a probability distribution associated with X, Y and Z. The *conditional mutual information* of X and Y given Z is defined:

$$\begin{aligned}\mathcal{I}[\mu](X; Y|Z) &= \mathcal{H}[\mu](X|Z) - \mathcal{H}[\mu](X|Y, Z) \\ &= \mathcal{H}[\mu](Y|Z) - \mathcal{H}[\mu](Y|X, Z)\end{aligned}$$

Quantitative information flow. Let P be a program. Let $\mathbb{H}$ be the set of valuations of high variables, let $\mathbb{L}$ be the set of valuations of low variables. Let μ be a probability distribution over $\mathbb{H} \times \mathbb{L}$. Let h be a valuation in $\mathbb{H}$, and let l be a valuation in $\mathbb{L}$. The expression $\mu(H = h, L = l)$ denotes the probability that the valuation of the high variables (variables in H) is h and the valuation of the low variables (variables in L) is l.

Let P be a program. Let $\mathcal{T}$ range over traces of P. Given a program (by definition deterministic) and an observation-based strategy ι, we can extend μ to traces by:

$$\mu_\iota(\mathcal{T} = \sigma) = \sum_{h \in \mathbb{H},\ l \in \mathbb{L},\ P(h,l,\iota) = \sigma} \mu(H = h, L = l),$$

where $P(h, l, \iota)$ is the observation sequence obtained by executing the program P with initial valuations given by h and l, and input variables set at each step by strategy ι.

The (Shannon-entropy-based) quantitative information flow $SE[\mu](P)$ is defined:

$$\begin{aligned}SE[\mu](P) &= \sup_\iota \mathcal{I}[\mu_\iota](\mathcal{T}; H|L) \\ &= \sup_\iota (\mathcal{H}[\mu_\iota](H|L) - \mathcal{H}[\mu_\iota](H|\mathcal{T}, L))\end{aligned}$$

Note that according to this definition, the opponent has the control of the input variables, and can choose new input values based on the observations he or she sees. The definition considered e.g. in [9], [11] is a special case of the definition presented here, where there are no input variables over which the opponent has control, and the output is just a single value (as opposed to a sequence of values). As in [9], [10], we obtain that requiring the information flow quantity to be 0 amounts to requiring noninterference. The proof is a variant of the proof in [9].

Theorem 1: Let μ be a probability distribution such that for all $h \in \mathbb{H}$ and for all $l \in \mathbb{L}$, we have that $\mu(H = h, L = l) > 0$. A program P is noninterferent iff $SE[\mu](P) = 0$.

D. Decision Problems

We present the three decision problems we consider.

1) *Program quantitative information flow (PQIF) bounding problem.* Given a program P, a probability distribution μ over $\mathbb{H}$ and $\mathbb{L}$, a rational number d, the *program quantitative information flow (PQIF) bounding problem* is to decide whether $SE[\mu](P) \geq d$. In other words, the question asks whether there is an input sequence for which the conditional mutual information is at least d.
2) *QIF memoryless synthesis problem.* Given a partial program M, let $\mathcal{P}$ be the set of programs allowed by M. Given a partial program M, a probability distribution μ over $\mathbb{H}$ and $\mathbb{L}$, a rational number d, the *QIF memoryless synthesis problem* is to decide whether for every program P in $\mathcal{P}$ we have $SE[\mu](P) \geq d$. In other words, the question asks whether for every memoryless refinement (choosing one transition for each choice) the answer to the PQIF problem is yes.
3) *QIF synthesis problem.* Given a partial program M, let $\overline{\mathcal{P}}$ be the set of programs allowed by M by possibly adding more low variables. Given a partial program

M, a probability distribution μ over $\mathbb{H}$ and $\mathbb{L}$, a rational number d, The *QIF synthesis problem* is to decide whether for every program P in $\overline{\mathcal{P}}$ we have $SE[\mu](P) \geq d$. In other words, the question asks whether for every refinement of the program M (with possibly adding more variables) the answer to the PQIF problem is yes.

III. Partial Information Non-deterministic and Deterministic Systems

In this section we present the notions of partial information non-deterministic and deterministic systems, and establish equivalence with the partial program and program models.

Partial information non-deterministic and deterministic systems. A *partial information non-deterministic system* (PINS) is a tuple $G = \langle L, \Sigma, \Delta, \mathcal{O}, \gamma, I \rangle$, where L is a finite set of states, Σ is a finite alphabet of input letters, $\Delta \subseteq L \times \Sigma \times L$ is a set of labeled transitions, $\mathcal{O}$ is a finite set of observations, $\gamma : \mathcal{O} \to 2^L \setminus \emptyset$ maps each observation to the set of states that it represents, and I is the set of initial states. We require the following two properties on G: (i) for all $\ell \in L$ and all $\sigma \in \Sigma$, there exists $\ell' \in L$ such that $(\ell, \sigma, \ell') \in \Delta$; and (ii) the set $\{\gamma(o) \mid o \in \mathcal{O}\}$ partitions L. A *partial information deterministic system* (PIDS) is a special case of PINS where the transition relation is deterministic, i.e, for all $\ell \in L$ and $\sigma \in \Sigma$, the set $\{\ \ell' \mid (\ell, \sigma, \ell') \in \Delta\ \}$ is singleton. In other words, the transition relation can be interpreted as a function $\Delta : L \times \Sigma \to L$.

Two players for PINS. A PINS is a dynamic system that evolves by the interaction of two players: Player 1 chooses the input letters and Player 2 resolves the non-determinism. Player 1 will have partial-information as Player 1 will only have access to the observations and not the precise state, i.e., only the observation sequence will be *visible* to Player 1, not the precise state sequence.

Plays. In a PINS, in each turn, Player 1 chooses a letter in Σ, and Player 2 resolves nondeterminism by choosing the successor state. A *play* in G is an infinite sequence $\pi = \ell_0\sigma_0\ell_1 \ldots \sigma_{n-1}\ell_n\sigma_n \ldots$ such that (i) $\ell_0 \in I$ (i.e., ℓ_0 is an initial state), and (ii) for all $i \geq 0$, we have $(\ell_i, \sigma_i, \ell_{i+1}) \in \Delta$. The set of plays in G is noted $\mathsf{Plays}(G)$. The *prefix up to* ℓ_n of the play π is denoted by $\pi(n)$; its *length* is $|\pi(n)| = n + 1$; and its *last element* is $\mathsf{Last}(\pi(n)) = \ell_n$. The *observation sequence* of π is the unique infinite sequence $\gamma^{-1}(\pi) = o_0\sigma_0o_1 \ldots \sigma_{n-1}o_n\sigma_n \ldots$ such that for all $i \geq 0$, we have $\ell_i \in \gamma(o_i)$. Similarly, the *observation sequence* of $\pi(n)$ is the prefix up to o_n of $\gamma^{-1}(\pi)$. The set of infinite plays in G is denoted $\mathsf{Plays}(G)$, and the set of corresponding finite prefixes is denoted $\mathsf{Prefs}(G)$. A state $\ell \in L$ is *reachable* in G if there exists a prefix $\rho \in \mathsf{Prefs}(G)$ such that $\mathsf{Last}(\rho) = \ell$. The *knowledge* associated with a finite observation sequence $\tau = o_0\sigma_0o_1\sigma_1 \ldots \sigma_{n-1}o_n$ is the set $\mathsf{K}(\tau)$ of states in which a play can be after this sequence of observations, that is, $\mathsf{K}(\tau) = \{\mathsf{Last}(\rho) \mid \rho \in \mathsf{Prefs}(G) \text{ and } \gamma^{-1}(\rho) = \tau\}$. In other words, the larger is the knowledge set, the larger is the uncertainty about the knowledge about the precise state of the system. The following lemma gives the inductive construction of the knowledge with observation sequence. For $\sigma \in \Sigma$ and $s \subseteq L$, let $\mathsf{Post}^G_\sigma(s) = \{\ell' \in L \mid \exists \ell \in s : (\ell, \sigma, \ell') \in \Delta\}$ denote the set of possible successors of states in s.

Lemma 1: Let $G = \langle L, \Sigma, \Delta, \mathcal{O}, \gamma, I \rangle$ be a PINS. For $\sigma \in \Sigma$, $\ell \in L$, and $\rho, \rho' \in \mathsf{Prefs}(G)$ with $\rho' = \rho \cdot \sigma \cdot \ell$, let $o_\ell \in \mathcal{O}$ be the unique observation such that $\ell \in \gamma(o_\ell)$. Then $\mathsf{K}(\gamma^{-1}(\rho')) = \mathsf{Post}^G_\sigma(\mathsf{K}(\gamma^{-1}(\rho))) \cap \gamma(o_\ell)$.

Strategies. Strategies are recipes for the players to extend plays. Formally, a *strategy* in G for Player 1 is a function $\alpha : \mathsf{Prefs}(G) \to \Sigma$. A strategy α for Player 1 is *observation-based* if for all prefixes $\rho, \rho' \in \mathsf{Prefs}(G)$, if $\gamma^{-1}(\rho) = \gamma^{-1}(\rho')$, then $\alpha(\rho) = \alpha(\rho')$. Observation-based strategies formalizes the partial-information (or that only the observations are visible) for Player 1. In the sequel, we are interested in the existence of observation-based strategies for Player 1. A *strategy* in G for Player 2 is a function $\beta : \mathsf{Prefs}(G) \times \Sigma \to L$ such that for all $\rho \in \mathsf{Prefs}(G)$ and all $\sigma \in \Sigma$, we have $(\mathsf{Last}(\rho), \sigma, \beta(\rho, \sigma)) \in \Delta$. A strategy β for Player 2 is memoryless if it is independent of the history and depends only on the current state and input letter, i.e., for all $\rho, \rho' \in \mathsf{Prefs}(G)$ if $\mathsf{Last}(\rho) = \mathsf{Last}(\rho')$, then for all σ we have $\beta(\rho, \sigma) = \beta(\rho', \sigma)$. A memoryless strategy for Player 2 can be interpreted as a function $\beta : L \times \Sigma \to L$. We denote by $\mathcal{A}_G$, $\mathcal{A}^O_G$, $\mathcal{B}_G$, and $\mathcal{B}^M_G$ the set of all Player-1 strategies, the set of all observation-based Player-1 strategies, the set of all Player-2 strategies, and the set of all Player 2 memoryless strategies in G, respectively.

Strategies for PIDS. Observe that in a PIDS, since the transition function is deterministic there is only one strategy for Player 2, and in PIDS it is only Player 1 that has choice of strategies.

Outcome. The *outcome* of two strategies α (for Player 1) and β (for Player 2) in G is the play $\pi = \ell_0\sigma_0\ell_1 \ldots \sigma_{n-1}\ell_n\sigma_n \ldots \in \mathsf{Plays}(G)$ such that for all $i \geq 0$, we have $\sigma_i = \alpha(\pi(i))$ and $\ell_{i+1} = \beta(\pi(i), \sigma_i)$. This play is denoted $\mathsf{outcome}(G, \alpha, \beta)$. The *outcome set* of the strategy α for Player 1 in G is the set $\mathsf{Outcome}_i(G, \alpha)$ of plays π such that there exists a strategy β for Player 2 with $\pi = \mathsf{outcome}(G, \alpha, \beta)$ The outcome sets for Player 2 are defined symmetrically.

Objectives. An *objective* for G in general is a set ϕ of the plays. that is, $\phi \subseteq \mathsf{Plays}(G)$. A play $\pi = \ell_0\sigma_0\ell_1 \ldots \sigma_{n-1}\ell_n\sigma_n \ldots \in \mathsf{Plays}(G)$ *satisfies* the objective ϕ, denoted $\pi \models \phi$, if $\pi \in \phi$. In this work we consider the *bounded knowledge objective*. The bounded knowledge

objective consists of a number $q \in \mathbb{N}$ and requires that the knowledge set is at most of size q at some point. Formally, the bounded knowledge objective $\phi_K(q)$ is defined as the following set: $\{ \ell_0\sigma_0\ell_1\sigma_1 \ldots \in \mathsf{Plays}(G) \mid \exists k \geq 0.\ |\mathsf{K}(\gamma^{-1}(\ell_0\sigma_1\ell_1 \ldots \ell_k))| \leq q\}$. Observe that by definition, for objectives $\phi_K(q)$, if $\pi \models \phi_K(q)$ and $\gamma^{-1}(\pi) = \gamma^{-1}(\pi')$, then $\pi' \models \phi_K(q)$. We also distinguish the special case of *qualitative* bounded knowledge objective for the case when $q = 1$, i.e., the objective requires that at some point there is perfect or precise knowledge (i.e., the knowledge set is singleton, and hence the knowledge is perfect).

Winning and winning strategies. A strategy λ_i for Player i in G is *winning* for an objective ϕ if for all $\pi \in \mathsf{Outcome}_i(G, \lambda_i)$, we have $\pi \models \phi$. In this work we are interested in the existence of observation-based winning strategies for Player 1 for objectives $\phi_K(q)$. For the memoryless question we ask whether there is an observation-based strategy for Player 1 that is winning against all memoryless strategies for Player 2.

(Partial) Programs, PIDS*, and* PINS*:* In order to prove upper and lower bounds for decision problems concerning information flow in programs and partial programs, we will use the following correspondence between (partial) programs on one side, and PIDS and PINS on the other side.

Given a PINS $G = \langle L_G, \Sigma, \Delta, \mathcal{O}, \gamma, I_G\rangle$, we construct a partial program $P = \langle Q, H, L, I, \delta, q_0\rangle$ as follows. The set of control locations is L_G. The location q_0 is the initial location. The program has one high variable h which ranges over the set L_G. There is one low variable which ranges over the set $\mathcal{O}$. The program will have one input variable ranging over the set Σ. The transition function δ encodes the transition function Δ. The partial program obtained from a PINS G is denoted by $\rho(G)$. We remark that by using the same construction on a PIDS we obtain a (deterministic) program.

Given a partial program $M = \langle Q, H, L, I, \delta, q_0\rangle$, we construct a PINS $G = \langle L_G, \Sigma, \Delta, \mathcal{O}, \gamma, I_G\rangle$ as follows. A state in the set L_G of states consists of a control location of M, and of valuations of high, low, and input variables of M. The set Σ is the set of valuations of input variables. The set $\mathcal{O}$ is the set of valuations of low variables. The function γ maps each observation (i.e. a valuation of low variables) to the subset of L_G which has the same valuation of low variables. The set I_G of initial states is the set of states corresponding to the initial location q_0 of P. The function Δ intuitively combines both the program and the environment transition in the transition system $\mathsf{Tr}(M)$. Let (q, g, a, q') be a transition of M. There is a program transition from a state s to a state s' labeled by $\sigma \in \Sigma$ if and only if (i) the location of M in s is q and the location of M in s' is q', (ii) the valuations of input variables in s' are exactly those given in Σ (iii) the guard g evaluates to true in high and low variables in s and input variables in σ (i.e. the new input is considered), and (iv) the valuation of low variables of M in s', and is obtained from the valuation of low and high variables in s and input in σ by performing the operation a. The PINS obtained from a partial program M is denoted by $\xi(M)$. We remark that by using the same construction on a (deterministic) program, we obtain a PIDS.

IV. LOWER BOUNDS

In this section we show that the decision problems of whether there is an observation-based winning strategy for Player 1 for qualitative bounded knowledge objectives is EXPTIME-hard for PINS and PSPACE-hard for PIDS. The EXPTIME and PSPACE lower bound proofs will follow from reductions of membership problem for polynomial space Alternating and Deterministic Turing machines, respectively. We first recall the definitions of Alternating and Deterministic Turing Machines.

Alternating Turing Machine. An *Alternating Turing machine* (ATM) is a tuple $M = \langle Q, q_0, g, \Sigma_i, \Sigma_t, \delta, F\rangle$ where:

- Q is a finite set of control states;
- $q_0 \in Q$ is the initial state;
- $g : Q \to \{\wedge, \vee\}$;
- $\Sigma_i = \{0, 1\}$ is the input alphabet;
- $\Sigma_t = \{0, 1, 2\}$ is the tape alphabet and 2 is the *blank* symbol;
- $\delta \subseteq Q \times \Sigma_t \times Q \times \Sigma_t \times \{-1, 1\}$ is a transition relation; and
- $F \subseteq Q$ is the set of accepting states.

We say that M is a *polynomial space* ATM if for some polynomial $p(\cdot)$, the space used by M on any input word w is bounded by $p(|w|)$. An Alternating Turing Machine is *deterministic* if the transition relation δ is deterministic. We write ATM for Alternating Turing Machines, and DTM for Deterministic Turing Machines. For details of ATM and DTM see [18], [19].

The membership problem. Without loss of generality, we make the hypothesis that the initial control state of the machine is a $\vee$-state and that transitions link $\vee$-state to $\wedge$-state and vice versa. A word w is *accepted* by an ATM M if there exists a run tree of M on w whose all leaf nodes are accepting configurations (see [19] for details). The AND-OR graph of the polynomial space ATM (M, p) on the input word $w \in \Sigma^*$ is $G(M, p) = \langle S_\vee, S_\wedge, s_0, \Rightarrow, R\rangle$ where

- $S_\vee = \{(q, h, t) \mid q \in Q,\ g(q) = \vee, 1 \leq h \leq p(|w|)$ and $t \in \Sigma_t^{p(|w|)}\}$;
- $S_\wedge = \{(q, h, t) \mid q \in Q,\ g(q) = \wedge, 1 \leq h \leq p(|w|)$ and $t \in \Sigma_t^{p(|w|)}\}$;
- $s_0 = (q_0, 1, t)$ where $t = w.\Sigma_t^{p(|w|)-|w|}$;
- $((q_1, h_1, t_1), (q_2, h_2, t_2)) \in \Rightarrow$ iff there exists $(q_1, t_1(h_1), q, \gamma, d) \in \delta$ such that $q_2 = q$, $h_2 = h_1 + d$, $t_2(h_1) = \gamma$ and $t_2(i) = t_1(i)$ for all $i \neq h_1$;
- $R = \{(q, h, t) \in S_\vee \cup S_\wedge \mid q \in F\}$.

A word w is *accepted* by (M, p) iff the set R is *alternately* reachable (reachable in AND-OR graph sense) in $G(M, p)$ (see [19] for details related to AND-OR graphs and reachability in AND-OR graphs). The *membership problem* is to decide if a given word w is accepted by a given polynomial space ATM (M, p). The membership problem is EXPTIME-hard for ATM [19] and PSPACE hard for DTM [18].

Idea of the reduction. We first start with the main idea of the reduction. Given a polynomial space ATM M and a word w, we construct a PINS of size polynomial in the size of (M, w) to simulate the execution of M on w. Player 1 makes choices in $\vee$-states and Player 2 makes choices in $\wedge$-states. Furthermore, Player 1 is responsible for maintaining the symbol under the tape head. The objective for Player 1 is to reach an accepting configuration of the ATM, and once an accepting configuration is reached we will ensure that the qualitative bounded knowledge objective is satisfied.

Each turn proceeds as follows. In an $\vee$-state, by choosing a letter (t, a) in the alphabet of the PINS, Player 1 reveals (i) the transition t of the ATM that he has chosen (this way he also reveals the symbol that is currently under the tape head) and (ii) the symbol a under the next position of the tape head. If Player 1 lies about the current or the next symbol under the tape head, he should lose for the objective (objective not satisfied), otherwise the PINS proceeds. The machine is now in an $\wedge$-state and Player 1 has no choice: he announces a special symbol ϵ and Player 2, by resolving nondeterminism on ϵ, chooses a transition of the Turing machine which is compatible with the current symbol under the tape head revealed by Player 1 at the previous turn. The state of the ATM is updated and the PINS proceeds. The transition chosen by Player 2 is visible in the next state of the PINS and so Player 1 can update his knowledge about the configuration of the ATM. Player 1 wins whenever an accepting configuration of the ATM is reached, that is w is accepted. In other words, when an accepting configuration is reached, the state is revealed by a unique observation and then Player 1 has the perfect knowledge.

Two difficulties. There are two main difficulties in the reduction. We describe the difficulties and the solutions to overcome them below. Then we present the formal reduction.

Difficulty 1: Prevention of cheating. We say that Player 1 *cheats* if the player does not faithfully simulate the Turning machine and announces wrong content of the tape head. The first difficulty is to ensure that Player 1 loses when he announces a wrong content of the cell under the tape head. As the number of configurations of the polynomial space ATM is exponential, we cannot directly encode the full configuration of the ATM in the states of the game. To overcome this difficulty, we use the power of partial information as follows. Initially, Player 2 chooses a position k, $1 \leq k \leq p(|w|)$, on the tape: this number as well as the symbol $\sigma \in \{0, 1, 2\}$ that lies in the tape cell number k is maintained all along the game in the non-observable portion of the PINS states. The pair (σ, k) is thus private to Player 2 and invisible to Player 1. Hence, at any point in the game, Player 2 can check whether Player 1 is lying when announcing the content of cell number k, and go to a sink state if Player 1 cheats (no other states can be reached from there). Since Player 1 does not know which cell is monitored by Player 2 (k is private), to avoid losing, he should not lie about any of the tape cells and thus he should faithfully simulate the machine. Then, he wins the PINS for the knowledge objective if and only if the ATM accepts the word w.

Difficulty 2: One cheating is allowed. The second difficulty is that one cheating is allowed for Player 1 in the construction described above. Suppose Player 1 cheats for a tape cell j and by doing so reaches the accepting configuration in all other copies, other than tape cell j. Since Player 1 knows the cell where he is cheating, if all other runs corresponding to other tape cells reaches the accepting configuration, Player 1 can infer perfect knowledge in the end. To prevent this we duplicate the above reduction, and there is a copy of each state. If an accepting configuration is reached, then the perfect observation is revealed. If Player 1 cheats, then a *sink* state is reached, and then the observation is not revealed. Since there is a duplicate copy for sink states, it follows that if Player 1 cheats, then he cannot have perfect knowledge. It follows that Player 1 wins in the PINS if and only if the ATM accepts the word w.

We now present the details of the reduction of the hardness proof. For sake of clarity we first present the reduction that takes care of difficulty 1 (that is prevention of cheating) and the construction for duplicating is presented after that.

Reduction. Given a polynomial space ATM (M, p), with $M = \langle Q, q_0, g, \Sigma_i, \Sigma_t, \delta, F\rangle$ and a word w, we construct the following game structure $G_{M,p,w} = \langle L, \Sigma, \Delta, \mathcal{O}, \gamma, I\rangle$, where:

- The set of states $L = \{(\mathsf{sink}, i) \mid 1 \leq i \leq p(|w|)\} \cup L_1 \cup L_2$ where: $L_1 = (\delta \cup \{-\}) \times Q \times \{1, \ldots, p(|w|)\} \times \{1, \ldots, p(|w|)\} \times \Sigma_t$. A state (t, q, h, k, σ) consists of a transition $t \in \delta$ of the ATM chosen by Player 2 at the previous round or $-$ if this is the first round where Player 1 plays, the current control state q of M, the position h of the tape head, the pair (k, σ) such that the k-th symbol of the tape is σ, this pair (k, σ) will be kept invisible for Player 1. $L_2 = Q \times \{1, \ldots, p(|w|)\} \times \Sigma_t \times \{1, \ldots, p(|w|)\} \times \Sigma_t$. A state $(q, h, \gamma, k, \sigma)$ consists of q, h, k, σ as in L_1 and γ is the symbol that Player 1 claims to be under the tape head. The objective for Player 1 will be to reach a state $\ell \in L$ associated with an accepting control state of M, and once such a state is reached then Player 1 can observe the precise state.
- $I = \{(-, q_0, 1, k, \sigma) \mid$ where (i) $1 \leq k \leq p(|w|)$ and (ii) $\sigma = w(k)$ if $1 \leq k \leq |w|$ and $\sigma = 2$ otherwise

}. In other words, the initial state describes that tape cell is being monitored by Player 2, and is invisible to Player 1 (unless an accepting state is reached).

- $\Sigma = \{\epsilon\} \cup (\delta \times \Sigma_t)$.
- The transition relation Δ contains the following sets of transitions:
  - The *sink transitions* that contains transitions $((\mathsf{sink}, i), \sigma, (\mathsf{sink}, i))$ for all $\sigma \in \Sigma$ and $1 \leq i \leq p(|w|)$. In other words, when a sink state is entered, the PINS stays there forever.
  - We have the following transitions for L_1 states:
    1) $L_{1.1}$ that contains transitions $(\ell_1, \epsilon, (\mathsf{sink}, i))$ for all $\ell_1 \in L_1$ and i is the same as the fourth component of ℓ_1;
    2) $L_{1.2}$ that contains transitions $((t, q, h, k, \sigma), ((q_1, \gamma_1, q_2, \gamma_2, d), \gamma_3), (\mathsf{sink}, k))$ where $q_1 \neq q$ or $\neg(1 \leq h + d \leq p(|w|))$;
    3) $L_{1.3}$ contains the transitions $((t, q, h, k, \sigma), ((q_1, \gamma_1, q_2, \gamma_2, d), \gamma_3), (\mathsf{sink}, k))$ where $h = k \wedge \gamma_1 \neq \sigma$ or $h + d = k \wedge \gamma_3 \neq \sigma$; and
    4) $L_{1.4}$ contains the transitions $((t, q, h, k, \sigma), ((q_1, \gamma_1, q_2, \gamma_2, d), \gamma_3), (q_2, h + d, \gamma_3, k, \sigma'))$ such that $q = q_1$, $1 \leq h + d \leq p(|w|)$, $h = k \rightarrow (\gamma_1 = \sigma \wedge \sigma' = \gamma_2)$, and $h \neq k \rightarrow \sigma' = \sigma$.

    Those transitions are associated with states of the PINS where Player 1 chooses a transition of the ATM to execute (if he proposes ϵ, the PINS evolves to a sink state, see $L_{1.1}$). The transition proposed by Player 1 should be valid for the current control state of the ATM and the head should not exit the bounded tape after execution of the transition by the ATM, otherwise the PINS evolves to a sink state, see $L_{1.2}$. When choosing a letter, Player 1 also reveals the current letter under the tape head (given by the transition) as well as the letter under the next position of the tape head. If one of those positions is the one that is monitored by Player 2, the game evolves to a sink state in case Player 1 lies, see $L_{1.3}, L_{1.4}$.
  - We have the following transitions for L_2 states:
    1) $L_{2.1}$ contains the transitions $((q, h, \gamma_1, k, \sigma), \epsilon, ((q_1, \gamma_2, q_2, \gamma_3, d), q_3, h + d, k, \sigma'))$ such that $q = q_1$, $q_2 = q_3$, $\gamma_1 = \gamma_2$, $1 \leq h + d \leq p(|w|)$, $h = k \rightarrow \sigma' = \gamma_3$, and $h \neq k \rightarrow \sigma' = \sigma$;
    2) $L_{2.2}$ contains the transitions $((q, h, \gamma_1, k, \sigma), \epsilon, (\mathsf{sink}, k))$ such that there does not exist a transition $(q, \gamma_1, q_1, \gamma_2, d) \in \delta$ with $1 \leq h + d \leq p(|w|)$; and
    3) $L_{2.3}$ contains the transitions $((q, h, \gamma_1, k, \sigma), (t, \gamma), (\mathsf{sink}, k))$ where $(t, \gamma) \in \Sigma \setminus \{\epsilon\}$.

    Those transitions are associated with states of the PINS where Player 2 chooses the next transition of the ATM to execute. Player 1 should play ϵ otherwise the game goes to the sink state (see $L_{2.3}$). Also the game goes to the sink state if there is no valid transition to execute in the ATM (see $L_{2.2}$). In the other cases, when Player 1 proposes ϵ, Player 2 chooses a valid transition by resolving nondeterminism. The copy of the monitored cell is updated if necessary.
- The set of observations $\mathcal{O} = \{\mathsf{sink}\} \cup \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{O}_3 \cup \mathcal{O}_4$ is as follows:
  1) $\mathcal{O}_1 = \{(t, q, h) \mid \exists (t, q, h, k, \sigma) \in L_1, q \notin F\}$,
  2) $\mathcal{O}_2 = \{(q, h, \gamma) \mid \exists (q, h, \gamma, k, \sigma) \in L_2, q \notin F\}$,
  3) $\mathcal{O}_3 = \{(t, q, h, k, \sigma) \mid (t, q, h, k, \sigma) \in L_1, q \in F\}$, and
  4) $\mathcal{O}_4 = \{(q, h, \gamma, k, \sigma) \mid (q, h, \gamma, k, \sigma) \in L_2, q \in F\}$.
- The observation mapping γ is defined as follows:
  1) $\gamma(\mathsf{sink}) = \{(\mathsf{sink}, i) \mid 1 \leq i \leq p(|w|)\}$,
  2) for all $(t, q, h) \in \mathcal{O}_1$, $\gamma(t, q, h) = \{(t, q, h, k, \sigma) \in L_1\}$,
  3) for all $(q, h, \gamma) \in \mathcal{O}_2$, $\gamma(q, h, \gamma) = \{(q, h, \gamma, k, \sigma) \in L_2\}$,
  4) for all $(t, q, h, k, \sigma) \in \mathcal{O}_3$, $\gamma(t, q, h, k, \sigma) = \{(t, q, h, k, \sigma)\}$, and
  5) for all $(q, h, \gamma, k, \sigma) \in \mathcal{O}_4$, $\gamma(q, h, \gamma, k, \sigma) = \{(q, h, \gamma, k, \sigma)\}$.

  Note that the observation mapping ensures that for every observation in $\mathcal{O}_3$ and $\mathcal{O}_4$ there is a unique state for the observation (i.e., perfect knowledge).

It follows that Player 1 has an observation-based strategy to ensure that in all copies of (k, σ) an accepting state in F is reached iff the word w is accepted by the polynomial space ATM (M, p). To complete the reduction to PINS with qualitative bounded knowledge objective we need the duplication step. Consider the construction $G_{M,p,w} = \langle L, \Sigma, \Delta, \mathcal{O}, \gamma, I \rangle$, presented above, we construct $\overline{G}_{M,p,w} = \langle \overline{L}, \Sigma, \overline{\Delta}, \overline{\mathcal{O}}, \overline{\gamma}, \overline{I} \rangle$ that contains duplication of $G_{M,p,w}$ as follows:

- $\overline{L} = L \times \{1, 2\}$ (i.e. there are two copies 1 and 2 of L);
- $\overline{I} = I \times \{1, 2\}$;
- $\overline{\Delta} = \{((\ell, i), \sigma, (\ell', i)) \mid \sigma \in \Sigma, \ell \in L, i \in \{1, 2\}, (\ell, \sigma, \ell') \in \Delta\}$;
- $\overline{\mathcal{O}} = \{\mathsf{sink}\} \cup \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{O}_3 \times \{1, 2\} \cup \mathcal{O}_4 \times \{1, 2\}$.
- The observation mapping is as follows:
  1) $\gamma(\mathsf{sink}) = \{((\mathsf{sink}, i), j) \mid 1 \leq i \leq p(|w|), 1 \leq j \leq 2\}$,
  2) for all $(t, q, h) \in \mathcal{O}_1$, $\gamma(t, q, h) = \{((t, q, h, k, \sigma), j) \in L_1 \times \{1, 2\}\}$,

3) for all $(q,h,\gamma) \in \mathcal{O}_2$, $\gamma(q,h,\gamma) = \{((q,h,\gamma,k,\sigma),j) \in L_2 \times \{1,2\}\}$,
4) for all $((t,q,h,k,\sigma),j) \in \mathcal{O}_3 \times \{1,2\}$, $\gamma((t,q,h,k,\sigma),j) = \{((t,q,h,k,\sigma),j)\}$, and
5) for all $((q,h,\gamma,k,\sigma),j) \in \mathcal{O}_4 \times \{1,2\}$, $\gamma((q,h,\gamma,k,\sigma),j) = \{((q,h,\gamma,k,\sigma),j)\}$.

Again observe that for every observation in $\mathcal{O}_3 \times \{1,2\}$ and $\mathcal{O}_4 \times \{1,2\}$ there is a unique state for the observation.

The above construction consists of two copies of the $G_{M,p,w}$ construction, but when a state in F is reached, the identity of the copy is revealed as well. Hence the perfect knowledge can be obtained by Player 1 iff in all copies an accepting state is reached. It follows that Player 1 has an observation-based winning strategy for the qualitative bounded knowledge objective iff the word w is accepted by the polynomial space ATM (M,p). Since our reduction is polynomial the EXPTIME-hardness follows for qualitative bounded knowledge problem for PINS.

Remark 1. We observe that our previous reduction preserves that if the transition function δ of the Turing Machine M is deterministic, then the transition function $\overline{\Delta}$ of $\overline{G}_{M,p,w}$ is also deterministic. Hence the above reduction for DTM gives us a PIDS and from the PSPACE-hardness of the membership problem for DTM we obtain the PSPACE-hardness for PIDS. Hence we have the following theorem.

Theorem 2 (Lower bounds): The decision problem of whether Player 1 has an observation-based winning strategy for qualitative bounded knowledge objectives is EXPTIME-hard for PINS and PSPACE-hard for PIDS.

Lower bound for partial programs and programs. We now show how from the lower bound for PIDS and PINS we obtain the lower bounds for PQIF bounding, QIF memoryless synthesis and QIF synthesis problems.

Lemma 2 (Lower bound for PQIF): The PQIF bounding problem is PSPACE-hard.

Proof: Let us consider the PQIF bounding problem given by a program P, a probability distribution μ over $\mathbb{H}$ and $\mathbb{L}$, and a rational number d. Recall that the problem is to decide whether $SE[\mu](P) \geq d$.

We reduce the decision problem of whether Player 1 has an observation-based winning strategy for qualitative bounded knowledge objectives in PIDS to the PQIF bounding problem.

Given a PIDS G, we create an instance of the PQIF bounding problem as follows: we consider the program $P = \rho(G)$ (recall the function ρ defined in Section III) with H and L denoting the high and low variables of P, a probability distribution μ such that $\mathcal{H}[\mu](H|L) > 0$ (this is always possible as long as the PIDS G is such that there is more than one state corresponding to each initial observation), and a rational number d equal to $\mathcal{H}[\mu](H|L)$ (we can choose μ such that d is rational). As in Section II, let ι denote an observation-based strategy, $\mathcal{T}$ a set of sequences of observations and let μ_ι denote the probability distribution μ extended to traces.

A winning strategy α for Player 1 has the property that for all plays π corresponding to α, the knowledge set is a singleton. Given such a strategy α, let us consider a strategy ι obtained in a natural way from α. For all such strategies ι, $\mathcal{H}[\mu_\iota](H|\mathcal{T},L) = 0$. This is because as the knowledge set is a singleton after Player 1 plays according to α, after seeing the sequence σ (constructed by feeding inputs using ι to the program), the observer knows the value of H exactly. Therefore, the existence of a winning strategy for the qualitative bounded knowledge objective is equivalent to the existence of a strategy ι such that $\mathcal{H}[\mu_\iota](H|\mathcal{T},L) = 0$.

Recall that

$$SE[\mu](P) = \sup_{\iota}(\mathcal{H}[\mu](H|L) - \mathcal{H}[\mu_\iota](H|\mathcal{T},L)).$$

As $\mathcal{H}[\mu](H|L)$ does not depend on ι, we have that

$$SE[\mu](P) = \mathcal{H}[\mu](H|L) - \inf_{\iota}\mathcal{H}[\mu_\iota](H|\mathcal{T},L)).$$

If we set d to $\mathcal{H}[\mu](H|L)$, the question $SE[\mu](P) \geq d$ amounts to asking $\inf_\iota \mathcal{H}[\mu_\iota](H|\mathcal{T},L) \leq 0$. The fact that $\mathcal{H}[\mu_\iota](H|\mathcal{T},L)$ is always nonnegative, and the fact that the set of states is finite, imply that $\inf_\iota \mathcal{H}[\mu_\iota](H|\mathcal{T},L) \leq 0$ iff there exists an observation-based strategy ι of inputs such that $\mathcal{H}[\mu_\iota](H|\mathcal{T},L) = 0$. This completes the desired reduction. The reduction is polynomial and hence we obtain the PSPACE hardness result. ■

Corollary 1 (Lower bound for synthesis problems): The following assertions hold:

- The QIF memoryless synthesis problem is PSPACE-hard.
- The QIF synthesis problem is EXPTIME-hard.

Proof: We prove both assertions as follows.

- We observe that a deterministic program is a special case of partial program, and hence the PQIF bounding problem is a special case of the QIF memoryless synthesis problem, since given a deterministic program the QIF memoryless synthesis problem is the same as the PQIF bounding problem. Hence we obtain the PSPACE-hardness result.
- The straightforward extension of the proof of the previous lemma shows that QIF synthesis problem can be reduced to the qualitative bounded knowledge problem for PINS. Hence from the lower bound for PINS we obtain the EXPTIME-hardness result.

The desired result follows. ■

V. UPPER BOUNDS

In this section we show that for PIDS the quantitative information flow problem can be solved in PSPACE. We then show that with the PSPACE solution for PIDS, the memoryless problem for PINS can also be solved in PSPACE.

Also note that the quantitative information flow problem for PIDS is a special case of the memoryless problem for PINS. Along with the PSPACE hardness result for PIDS we obtain PSPACE-complete complexity for PIDS and memoryless problem for PINS. To prove our result we consider the unravelling of a PIDS as a computation tree for an observation-based strategy.

The computation tree. Given a PIDS G and an observation-based strategy α we construct the computation tree $T_{G,\alpha}$ as follows: we will refer to the vertices of the tree as nodes, and each node will represent the current knowledge of Player 1. Every node of the tree represent a set of states of G.

- The root is the set I of initial states.
- Consider a node x in the tree, and let the $s(x)$ denote the set of states of x. Let the observation and input letter sequence from the root to x be $\rho = o_0\sigma_0 o_1\sigma_1 \dots \sigma_{n-1}o_n$, and let $\sigma_n = \alpha(\rho)$ be the input letter specified by the strategy α for the sequence ρ. Let $s'(x) = \mathsf{Post}_{\sigma_n}(s(x))$ be the set of possible successor states from $s(x)$ given σ_n. Consider the partition of $s(x)$ by the observations as $s(x_1), s(x_2), \dots, s(x_k)$, i.e., for all $1 \leq i \leq k$ we have $s(x_i) \subseteq s'(x)$, $s(x_i)$ is non-empty and there exists an observation o such that $s(x_i) = s'(x) \cap \gamma(o)$. The children of x are $x_1, x_2, \dots, x_k$ and for $1 \leq i \leq k$ each x_i is labeled by $s(x_i)$.

The weight bounding problem for PIDS**.** Let w be a weight function that assigns to every knowledge set K the entropy $w(K)$ associated with the knowledge set. The weight function will have the monotonicity property that if we have two knowledge sets K_1 and K_2 and K_1 is more refined in terms of knowledge (i.e., $K_1 \subseteq K_2$), then the entropy for K_1 is lower than K_2, i.e., $w(K_1) \leq w(K_2)$. We will show that the PQIF bounding problem (for deterministic programs) reduces to the following *weight bounding* problem for PIDS: the input to the problem is a PIDS G, weight function w, and a number λ, and the questions asks whether there exists an observation-based strategy and a number k of steps, such that after k steps the sum of the entropy of the knowledge sets is at most λ. In other words, for an observation-based strategy α, and a level k, let $W_k(G, w, \alpha)$ be the sum of the entropy of level k of the computation tree $T_{G,\alpha}$. Let $W(G, w, \alpha) = \inf_{k\in\mathbb{N}} W_k(G, w, \alpha)$ and let $W(G, w) = \inf_{\alpha\in\mathcal{A}^O_G} W(G, w, \alpha)$. The weight bounding problem asks whether $W(G, w) \leq \lambda$. We will present a PSPACE upper bound for PIDS. For the PSPACE upper bound we first present the key monotonicity property.

Monotonicity property. Our upper bound proof will rely on a *monotonicity* property of the computation tree. Consider a node x of the computation tree, and let $x_1, x_2, \dots, x_k$ be the children nodes. Since we consider a PIDS it follows that the size of $s'(x)$ and $s(x)$ are the same, i.e., $|s'(x)| = |s(x)|$. Since the set $s'(x)$ is partitioned among the children nodes, it follows that $|s(x_i)| \leq |s(x)|$ for all $1 \leq i \leq k$. Hence it follows that along a path in the computation tree the knowledge set is non-increasing (i.e., the entropy is non-increasing). We now present the following lemma which is the key for the upper bound and the lemma will use the monotonicity property.

Lemma 3: Given a PIDS G, and a number λ, if there exists an observation-based strategy and a number k of steps, such that after k steps the sum of the entropy of the knowledge sets is at most λ, then there exists an observation-based strategy that within $\ell \leq 2^n$ steps ensures that the sum of the entropy of the knowledge sets after ℓ steps is at most λ, where n is the number of states of the PIDS.

Proof: The proof is as follows: consider an observation-based strategy α that achieves the objective in minimum number of steps. Consider the computation tree $T_{G,\alpha}$. Consider a path of length greater than 2^n, then there must be two nodes x_1 and x_2 in the path such that $s(x_1) = s(x_2)$, and hence by the monotonicity property the entropy does not change along the segment of the path between x_1 and x_2. Hence the segment can be collapsed without the change of entropy. Hence it follows that if there is an observation-based strategy to achieve the objective, then there is one to achieve the objective in at most 2^n steps. ■

The PSPACE **upper bound.** We present a non-deterministic polynomial space algorithm, and since NPSPACE is same as PSPACE [18], the desired PSPACE upper bound will follow. The non-deterministic polynomial space algorithm is as follows: it constructs an observation-based strategy on the fly, constructs the computation tree level by level, and only remembers the last level. We make the following observation: if we consider any level i of the tree and let the nodes be $x_1, x_2, \dots, x_m$ in level i, then $|\bigcup_{1\leq i\leq m} s(x_i)| = |I|$, i.e., the number of the states in the union of the sets of states is at most the size of I. This follows because in each step the number of states of a parent node is partitioned in the children node. It follows that for any level i, there can be at most n nodes, where n is the size of the state space of G. To construct the observation-based strategy on the fly, we need a counter to count upto 2^n steps (by Lemma 3 we require at most 2^n steps). We require only polynomial space in n to count upto 2^n (since we require $\log r$ bits to count upto r). The second useful observation is that every path in the computation tree represents a different observation sequence. Hence the on-the-fly strategy construction is achieved as follows: the algorithm remembers in memory the nodes x and the sets of states $s(x)$ associated with the node at the current level, and the update from the current level of the tree to the next level is achieved as follows:

1) First, the algorithm guesses the input letter for every node in the current level,
2) then the algorithm updates the children of the computation tree and thus constructs the new current level,

3) it erases the old level and keeps only the new level of the tree,
4) it checks whether the sum of the entropy of the nodes of the current level is at most λ, if Yes the algorithm stops with success of finding an observation-based strategy, else goes to the next step,
5) increment the counter, and if the counter reaches 2^n, then the algorithm stops with failure, if the counter is less than 2^n, then the algorithm goes to build the next level of the tree

Note that since in each level there are at most n nodes, each iteration is achieved in polynomial space. At any point, the algorithm remembers the current level of the computation tree which requires only polynomial space, along with the polynomial space for the counter. Hence we have a PSPACE algorithm for the entropy problem for PIDS. We obtain the following result.

Lemma 4: The weight bounding problem for PIDS can be decided in PSPACE.

The memoryless problem for partial-programs. The QIF memoryless synthesis for partial programs problem reduces to the following memoryless weight bounding problem for PINS (a straightforward extension of the reduction for deterministic programs): given a PINS we ask whether against every memoryless strategy for Player 2 there is an observation-based winning strategy for Player 1 for the weight bounding problem. We use our results for PIDS to obtain a PSPACE upper bound. The upper bound is obtained as follows: suppose there is a memoryless strategy for Player 2 against which there is no observation-based winning strategy for Player 1, then the algorithm guesses the memoryless strategy, and once the memoryless strategy is fixed in the PINS we obtain a PIDS and the verification can be achieved in PSPACE (by Lemma 4). Since the guess of the memoryless strategy is only polynomial in the size of the PINS (the strategy only needs to fix one transition for each state and input letter), we can view the algorithm as a NP algorithm (for the polynomial guess) with a PSPACE oracle for verification. Hence we have as $\text{NP}^{\text{PSPACE}}$ algorithm (NP algorithm with a PSPACE oracle) and since NP is a subclass of PSPACE we have a $\text{PSPACE}^{\text{PSPACE}}$ algorithm. It follows from [18] that $\text{PSPACE}^{\text{PSPACE}}$ is same as PSPACE, and hence we have a PSPACE upper bound.

Lemma 5: The memoryless weight bounding problem for PINS can be decided in PSPACE.

Reduction. We now present the reductions of PQIF bounding and QIF memoryless synthesis problems to the weight bounding problem of PIDS and memoryless weight bounding problem for PINS.

Lemma 6 (Upper bound for programs): The PQIF bounding problem is in PSPACE.

Proof: Let us consider the PQIF bounding problem. We are given a program P, a probability distribution μ over $\mathbb{H}$ and $\mathbb{L}$, and a rational number d, and the problem is to decide whether $SE[\mu](P) \geq d$. We reduce the problem to the weight bounding problem for PIDS: we construct a PIDS G, a number λ, and a weight function w such that $W(G, w) \leq \lambda$. Note however that we need a probabilistic version of the weight bounding problem, one which weighs the entropy of knowledge sets by the probability of observation sequences leading to them. For simplicity, we did not introduce probabilities to PIDS, however the proof of membership in PSPACE can be modified for the probabilistic case.

The PIDS G will be obtained as $\xi(P)$ defined in Section III. Given a set of states K of $\xi(P)$ the weight function $w(K)$ will be defined by

$$\sum_{h \in Sc(K)} \mu(H = h) \log \frac{1}{\mu(H = h)},$$

where $Sc(K)$ is the set of valuations of high variables that appear in states of K. It is easy to see that weight function defined in this way fulfills the monotonicity requirement, as it essentially corresponds to the entropy over H according to an observer.

It remains to set the value λ. We set it to $\mathcal{H}[\mu](H|L) - d$.

We now show that the construction defines a reduction, i.e. the answer to the instance of the PQIF bounding problem is the same as the answer to the constructed instance of the weight bounding problem for PIDS.

By definition,

$$SE[\mu](P) = \sup_{\iota}(\mathcal{H}[\mu_\iota](H|L) - \mathcal{H}[\mu_\iota](H|\mathcal{T})).$$

As $\mathcal{H}[\mu_\iota](H|L)$ does not depend on ι, we have that

$$SE[\mu](P) = \mathcal{H}[\mu](H|L) - \inf_{\iota} \mathcal{H}[\mu_\iota](H|\mathcal{T}).$$

Let us now analyze $\mathcal{H}[\mu_\iota](H|\mathcal{T})$. We have that

$$\mathcal{H}[\mu_\iota](H|\mathcal{T}) = \sum_{\mathcal{T}=\sigma} \mu_\iota(\mathcal{T} = \sigma) \cdot \mathcal{H}[\mu_\iota](H|\mathcal{T} = \sigma),$$

where

$$\mathcal{H}[\mu_\iota](H|\mathcal{T} = \sigma) = \sum_{H=h} \mu_\iota(H = h|\mathcal{T} = \sigma) \cdot \log \frac{1}{\mu_\iota(H = h|\mathcal{T} = \sigma)}.$$

Let us now consider an observation-based strategy α for Player 1 in the PIDS $\xi(P)$ constructed from a strategy ι for choosing inputs for P in the natural way.

Considering the definition of $W(G, w, \alpha)$ and our construction of the weight function w, we have that $\mathcal{H}[\mu_\iota](H|\mathcal{T}) = W(G, w, \alpha)$, for α chosen as above, as in both cases, we are considering the entropy of H after performing a number of steps according to the same strategy. We thus have

$$\begin{aligned} SE[\mu](P) &= \mathcal{H}[\mu](H|L) - \inf_\iota \mathcal{H}[\mu_\iota](H|\mathcal{T}, L) \\ &= \mathcal{H}[\mu](H|L) - \inf_\alpha W(G, w, \alpha). \end{aligned}$$

The inequality $SE[\mu](P) \geq d$ therefore holds if and only if $\inf_\alpha W(G, w, \alpha) \leq \mathcal{H}[\mu](H|L) - d$, or equivalently, $\inf_\alpha W(G, w, \alpha) \leq \lambda$. By definition, this implies that $W(G, w) \leq \lambda$, which concludes the reduction. The reduction is polynomial, and hence with the upper bound for the weight bounding problem for PIDS (Lemma 4) we obtain the desired bound. ■

Corollary 2 (Upper bound for memoryless synthesis): The QIF memoryless synthesis problem is in PSPACE.

Proof: The reduction of the QIF memoryless synthesis problem to the memoryless weight bounding problem for PINS is the straight forward extension of the reduction of the previous lemma. The PSPACE upper bound (Lemma 5) establishes the desired PSPACE upper bound. ■

We have the following result summarizing the upper and lower bound for PQIF bounding and QIF memoryless synthesis problem.

Theorem 3: The PQIF bounding and QIF memoryless synthesis problems are PSPACE-complete.

The bound for QIF synthesis problem. We have already established that the QIF synthesis problem is EXPTIME hard and we now argue that the problem can be solved in 2EXPTIME using standard subset construction technique. The 2EXPTIME upper bound is obtained as follows:

1) *The weight bounding problem for perfect information systems.* Perfect information systems are special case of PINS where Player 1 can have precise knowledge about the state space. The weight bounding problem for perfect information systems can be solved in EXPTIME by subset construction. The subset construction maintain sets of states which represent the frontier of the tree. Hence the weight bounding problem for perfect information systems can be reduced to a reachability problem on a exponential size AND-OR graphs. Since reachability in AND-OR graphs can be solved in linear time, it follows that for perfection information systems, the weight bounding problem can be solved in EXPTIME.
2) *From* PINS *to perfect information systems.* The result of Reif [14] shows how with subset construction a partial information game can be reduced to an equivalent exponential size perfect information games. The standard subset construction is easily adapted to construct an equivalent exponential size perfect information system from a PINS.

Combining the above two subset construction we obtain a 2EEXPTIME upper bound for the QIF synthesis problem. Thus we have a 2EXPTIME upper bound and EXPTIME lower bound, and a matching upper and lower bound for the problem remains open.

Theorem 4: The QIF synthesis problem is EXPTIME-hard and can be solved in 2EXPTIME.

VI. CONCLUSION

Brief summary: In this work we studied the computational complexity of verification and synthesis problems for quantitative information flow. We establish optimal complexity for verification and memoryless synthesis problems showing they are PSPACE complete, while the general synthesis problem is EXPTIME-hard and can be solved in 2EXPTIME.

Future work: There are many directions for future research. An open question is to find a matching upper and lower bound for the QIF general synthesis problem. If one identifies domains where the quantitative relaxations of noninterference are useful (such as analysis of browser plugins or Java midlets), there is a question of whether the algorithms developed in this paper can be useful. Our results show that both verification and synthesis problems have high computational complexity. Therefore, from a more practical point of view, there is a question about how to develop suitable abstraction-based (semi)-algorithms for these problems.

Acknowledgements. This work was partially supported by the ERC Advanced Grant QUAREM, the FWF NFN Grant S11402-N23 and S11407-N23 (RiSE), and a Microsoft faculty fellowship.

REFERENCES

[1] F. Schneider, Ed., *Trust in Cyberspace*. National Academy Press, 1999.

[2] S. Goodman and H. Lin, Eds., *Toward a Safer and More Secure Cyberspace*. National Academy Press, 2007.

[3] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[4] A. Sabelfeld and A. Myers, "Language-based information-flow security," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 5–19, 2003.

[5] A. Myers, "JFlow: Practical mostly-static information flow control," in *Proc. of POPL'99*, 1999, pp. 228–241.

[6] A. Sabelfeld and D. Sands, "Dimensions and principles of declassification," in *CSFW '05*, 2005, pp. 255–269.

[7] D. Denning, *Cryptography and data security*. Addison-Wesley, 1982.

[8] D. Clark, S. Hunt, and P. Malacaria, "A static analysis for quantifying information flow in a simple imperative language," *Journal of Computer Security*, vol. 15, no. 3, pp. 321–371, 2007.

[9] ——, "Quantified interference for a while language," *Electr. Notes Theor. Comput. Sci.*, vol. 112, pp. 149–166, 2005.

[10] H. Yasuoka and T. Terauchi, "Quantitative information flow - verification hardness and possibilities," in *CSF*, 2010, pp. 15–27.

[11] ——, "On bounding problems of quantitative information flow," in *ESORICS*, 2010, pp. 357–372.

[12] G. Smith, "On the foundations of quantitative information flow," in *FOSSACS*, 2009, pp. 288–302.

[13] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.

[14] J. Reif, "The complexity of two-player games of incomplete information," *Journal on Computer and System Sciences*, vol. 29, pp. 274–301, 1984.

[15] A. Solar-Lezama, R. Rabbah, R. Bodík, and K. Ebcioglu, "Programming by sketching for bit-streaming programs," in *PLDI*, 2005.

[16] A. Solar-Lezama, C. Jones, and R. Bodík, "Sketching concurrent data structures," in *PLDI*, 2008.

[17] M. Vechev, E. Yahav, and G. Yorsh, "Abstraction-guided synthesis of synchronization," in *POPL*, 2010.

[18] C. Papadimitriou, *Computational Complexity*. Reading, MA, USA: Addison-Wesley Publishing, 1994.

[19] A. K. Chandra, D. Kozen, and L. J. Stockmeyer, "Alternation." *J. ACM*, vol. 28, no. 1, pp. 114–133, 1981.