A Synchronous Model for Information Flow

Cătălin Dima, Constantin Enea, and Radu Gramatovici

Université Paris 12, 61 av. du Général de Gaulle, 94010 Créteil Cedex, France
University “Al. I. Cuza”, Str. General Berthlot 16, 740083, Iaşi, Romania
University of Bucharest, str. Academiei 14, 001014 Bucureşti, Romania

Abstract. We generalize the synchronous model of Wittbold & Johnson to a nondeterministic setting. Our model is of a game in which the high-level user and a low-level user play against a nondeterministic system. We investigate two notions of information flow in the new setting: a notion based on counting differently-observable high-level strategies and a notion based on the deductions that the low-level user makes on high-level activity while observing system behavior. The two notions focus on detecting different observations due to high-level choices, as means of communicating information from high to low. On the other hand, the system is supposed to be honest, and variations in observations induced by system nondeterminism are not considered as leaking information. We also give a decision procedure for finite-state systems. We relate our results to some known models of information flow: we show, for example, that bisimulation-based equivalence checking is too strong, since it labels variations in the observation due to system nondeterminism as information flow. We also show that a synchronous, trace-based statement of Generalized Noninterference does not properly capture information flow either. We give then a strategy-based version of Generalized Noninterference and prove it to be equivalent with our notion of information flow.

1 Introduction

Information flow is one of the main techniques that ensure confidentiality. There are several intuitions behind information flow, and they have been informally synthesized in statements about the (non-)dependence of high-level (“confidential”) activity and low-level (“public”) observation and/or deduction. These statements have produced a number of non-equivalent notions, all which try to emphasize on one of the terms within the statements. They resulted in a span of possibilistic models [GM82,Sut86,McC87,WJ90,ZL97,Man00a,McL94,RS99,FG95,Mar98] (to cite only a few) that classify information flow according to different aspects like observability (trace-based [Man00a,ZL97], bisimulation-based[FG95,FG00]), existence of high-level input or output events, deterministic or nondeterministic systems, composability [McL94], etc.

This paper has grew from the concern to approach information flow via a semantic model that would allow one to specify what the low-level user (Larry) is able to “deduce” of high-level (Harry’s) activity by observing system behaviors. Several notions of “deducibility” exists, starting with the synchronous framework in [WJ90], as well as bisimulation-based versions [FG95,FG00], but they do not rely on properly-defined notions of knowledge and deduction. [Cup93] investigates the use of Deontic Logic in connection with information flow, but without relating the approach to existing notions of information flow, and the recent work of [OH03] relates only to noninterference or separability. Also note [BS97], in which the authors do not investigate issues related to the sequential structure of computation.

On the other hand, we were looking for algorithms for checking whether a finite-state system leaks information. This direction is related to the so-called unwinding properties [Mil95,Man00b]. Note also some recent decidability results in [DRS05], related to the MAKS framework [Man00a].

Our approach starts from a system model very close to the one proposed by Wittbold & Johnson some 16 years ago. It is a synchronous model, in which both Harry and Larry have to interact with the system at each moment. Let us note that this strongly synchronous assumption does not disallow the
two users to choose not to provide any input – this situation can be modeled by introducing a distinct action \textit{idle} which, when issued by both users, allows the system to keep its state unchanged.

Why choosing a synchronous model and not an asynchronous one, as in most of the existing literature (including the recent CSP-based models)? The reason comes from the fact that, in our opinion, information flow is about \textit{common resources} – and \textit{time is such a common resource}. Synchronous models focus on a notion of global time, which seems to be more appropriate at least when dealing with information flow through timing. Asynchronous models focus on \textit{causality}, or, better said, on \textit{non-causality} between actions, which is a qualitative aspect of time. Therefore, in an asynchronous model, it is possible to have systems in which one of the components executes an arbitrary number of actions before another component can do anything; if we speak of Harry executing unboundedly many actions before Larry can see anything, this may lead to creating a covert channel through timing. Handling quantitative timing in an asynchronous model requires introducing a distinct \textit{clock tick} action for modeling time in each component, and inserting also special rules requiring time to flow \textit{synchronously} in each component \cite{FGM03}. But in a synchronous model quantitative timing comes for free!

Our model is of a \textit{game} played between the two users, Harry (high-level user), and Larry (low-level user), that both have only limited and disjoint access to system state. The game is also \textit{arbitrated} by the system – this is modeled by the fact that the transitions are \textit{nondeterministic}, and it’s eventually the system’s task to “solve” this nondeterminism. This is a feature of our model, as it allows us to \textit{synthesize controllers}, that is, system “strategies” or “patches” which may restrict system behavior such that no information flow be possible. This is one of the points in which our approach diverge from the approach of \cite{WJ90}.

It has been long agreed that information flow is about high-level \textit{decisions}. The possibility for Harry to \textit{choose} between several possible high-level inputs, and the fact that different choices lead to different \textit{observations} Larry can make, are the source of information flow. Similarly to \cite{WJ90}, we model Harry’s choices as \textit{strategies}. A strategy for Harry is a plan he has to interact with the system, \textit{function of} the system states that are accessible to him. This notion is also similar to the use, in \cite{FG00}, of high-level processes that interact with the system. A similar notion of strategy is also available for Larry.

A strategy Harry chooses may interact with a strategy Larry chooses and with system decisions for “solving nondeterminism”. In a deterministic setting, that is, in the absence of system nondeterminism, a strategy for Harry, interacting with the system and with a strategy for Larry would lead to a single run. Wittbold and Johnson define then information flow as the fact that two different strategies for Harry, when interacting with the same strategy for Larry, lead to two different runs that give two \textit{different observations for Larry}.

Nondeterminism complicates the task of defining information flow: the interaction between a strategy for Harry and a strategy for Larry may lead to different observations, due to different choices the system makes. The important point here is that we consider the system \textit{reliable}, that is, the eventual Trojan Horses can only act as Harry. This does not mean that our model is “unfair”: an unreliable system which may be subject of Trojan Horses attacks can be modeled in a deterministic framework in which Harry would have access to the entirety of the system state.

So, in order to separate system nondeterminism from information flow, we have to count \textit{classes of observations of runs} that are generated by the interaction between a strategy for Harry and a strategy for Larry. Similarly to \cite{Low04}, we define the covert channel capacity of a system as the cardinality of the set of distinct classes of observations of runs. Note however that in \cite{Low04,FG00} systems are modeled as CSP-like processes in which low-level users can only \textit{observe} the system, not being able to interact with it.
Another feature in our approach is the possibility to count only the strategies that do not put the system in a deadlocked state. In our model, a system state may have no continuation for a pair (high-level-input, low-level-input). We then take into account only admissible strategies that, at each moment, do not try to provoke such an impossible transition. Would this be another “unfair” assumption? Actually not: firstly, absent transitions may model situations in which the system has some Trojan Horse detector, and the Trojan Horse owner does not want his Trojan to be “captured stupidly”. Secondly, in language-based information flow models the interest is not always on the fact that Harry is virused, but rather on the leak of information about Harry’s activity when Harry is an honest agent (and hence always chooses correct inputs)! Thirdly, in any system, deadlocks can be made visible and our framework applies to input-total systems as well.

A final requirement we impose on systems is liveness, in the sense that at each moment at least one transition be enabled. We need this in order to differentiate deadlocks from successful computations that put the system in a “final state”. Final states are then modeled by sink states in which the system may loop with any input/output combination.

We then define “deducibility” in terms of the image that Larry, once he chooses a strategy to interact with the system, makes of the strategy Harry has chosen on its turn. The deduction Larry makes is dependent on the evolution of the system, which means that at each moment he only deduces some information on the part of Harry’s strategy that could have been chosen in order to produce the observations up to that moment. This form of “deducibility” is inspired by the knowledge of Dolev-Yao attackers against security protocols [DY83]. We believe this statement of information flow is new, and represents a “semantic model” for a logic of information flow, possibly related to the work of [Cup93]. We show that the definition based on computing the covert channel capacity is equivalent with this form of deducibility.

We then give an exponential-time decision procedure for checking whether a system has information flow in our setting. The algorithm idea is to represent classes of strategies equivalent w.r.t. low-level observations by finite classes of states. Note that our algorithm can detect any Trojan Horses, not only the finite-state ones like in [Mar98]. Our decision algorithm serves also for synthesizing system strategies for avoiding covert channels – that is, synthesizing system controllers that forbid certain low-high joint activities that lead to the leak of information.

We then compare our framework with Generalized Noninterference (GNI) [McC87]. We give first a translation of GNI into our synchronous framework by replacing interleaving – which is an asynchronous operation – with a combination of projections onto high and low level, which are natural operations in both asynchronous and synchronous models. We then show that “synchronous” GNI is incomparable with our notion of information flow. We give two examples of systems that satisfy synchronous GNI but do not satisfy our deducibility-based information flow property. The reason is that trace-based models do not properly capture system nondeterminism. Our results hold also for the case of H-input total systems, that is, systems in which, at any state, any high-level event is enabled. We also show that our definition of information flow does not imply the synchronous version of GNI.

We then give a strategy-based version of GNI, termed here Noninterference on Strategies, and show that it is equivalent with our definition of information flow. We do not give however strategy-based variants of the MAKS framework [Man00a]. The main reason for this choice is the fact that the MAKS framework is based on insertion or deletion of high-level events, which are asynchronous operations. But in a synchronous model, any event insertion is observable as a delay, hence one needs some restatement of the Basic Security Predicates of [Man00a] that utilizes replacement of high-level events instead of insertion/deletion. A complete comparison between our framework and Mantel’s is however a direction of further research.
Finally, we argue that our strategy-based definition of information flow is more appropriate than a bisimulation-based notion. We give an example of a system which has zero covert channel capacity, in our framework, but which does not satisfy the synchronous version of Bisimulation-based Nondeducibility on Compositions (BNDC) of [FG00]. And the source of this mismatch is that BNDC considers system decisions as a possible source of information leak, whereas in our framework we clearly separate the good (the system) from the evil (Harry).

The paper proceeds with a section presenting our system model and with our formalization of covert channel capacity. We then give the deducibility-based definition of information flow and show its equivalence with the covert-channel capacity based definition. The fourth section gives the decidability result and the fifth contains the discussion on the relationship with existing models. We also show how to use our decision procedure for synthesizing system strategies for avoiding covert channels. We end with a section with comments and possible directions for further research.

2 The game model

Throughout the paper we will assume that the set of events which may occur through the behavior of the system is \( E = H \cup L \), where \( H \) and \( L \) are the (nonnecessarily finite) sets of high-level, resp. low-level events. We assume that \( H \cap L = \emptyset \).

We will also use the notation \( S^n \) for the set of finite sequences over a set \( S \). The set of sequences over \( S \) of length \( n \) is denoted \( S^n \) while the set of sequences of length at most \( n \) is denoted by \( S^{\leq n} \).

Recall that an automaton is a tuple \( Aut = (Q, \Sigma, \delta, q_0) \) where \( Q \) is the set of states, \( \Sigma \) is the set of events, \( \delta \subseteq Q \times \Sigma \times Q \) is the transition relation and \( q_0 \) is the initial state (we consider here automata without final states). An automaton is finite state if \( Q \) and \( \Sigma \) are finite.

We will occasionally use \( \delta \) as a function \( \delta : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q) \), defined in the usual way:

\[
\delta(S, a) = \{ r \in Q \mid \exists q \in S, (q, a, r) \in \delta \}
\]

A run of length \( n \) in \( Aut \) is a sequence \( \rho = (q_{i-1} \xrightarrow{a_i} q_i)_{1 \leq i \leq n} \) such that \( (q_{i-1}, a_i, q_i) \in \delta \), for any \( 1 \leq i \leq n \), and \( q_0 \) is the initial state. We denote \( \text{len}(\rho) = n \) the length of the run. A run of infinite length is then an infinite sequence of transitions \( \rho = (q_{n-1} \xrightarrow{a_n} q_n)_{n \geq 1} \). We will also speak of \( \rho \) being an \( Aut \)-run. The set of runs for the automaton \( Aut \) is denoted \( \text{Runs}(Aut) \).

The prefix of length \( k \leq n \) of a run \( \rho = (q_{i-1} \xrightarrow{a_i} q_i)_{1 \leq i \leq n} \) is the run \( \rho[1..k] = (q_{i-1} \xrightarrow{a_i} q_i)_{1 \leq i \leq k} \). We will also denote \( \rho_1 \preceq \rho_2 \) when \( \rho_1 \) is a prefix of length \( k \) of \( \rho_2 \), for some \( k \leq \text{len}(\rho) \).

The immediate prefix of \( \rho \) is the run \( \rho[1..n-1] \); by abusing notation, we may denote the fact that \( \rho' \) is an immediate prefix of \( \rho \) as \( \rho = \rho' \xrightarrow{a} q \).

**Definition 1.** A game automaton (or system) is a tuple \( A = (Q, Q_H, Q_L, H, L, \delta, q_0, \chi, \lambda) \) where \( Q \) is the set of states, \( \chi : Q \to Q_H \) is the high-level projection of states, \( \lambda : Q \to Q_L \) is the low-level projection of states, \( H \) and \( L \) are the set of high-level, resp. low-level events, \( q_0 \) is the initial state and \( \delta \subseteq Q \times H \times L \times Q \) is the transition relation. The transition relation is also assumed to satisfy the following “liveness” property:

\[
\forall q \in Q \exists h \in H, l \in L, r \in Q \text{ s.t. } (q, h, l, r) \in \delta \tag{1}
\]

This is a usual nondeterministic automaton for a game between two players, call them Harry and Larry. As stated in the introduction, liveness is used for differentiating successful computations from deadlocked computations, in the following sense: instead of having finite successful computations, we
have computations which reach a “sink state”, that is, in which the system will loop forever with any inputs.

The **L-projection** and the **H-projection** of the transition function are:

\[ \delta^L_L = \{ (\lambda(q), l, \lambda(r)) \mid (q, h, l, r) \in \delta \text{ for some } h \in H \} \]
\[ \delta^L_H = \{ (\lambda(q), h, \lambda(r)) \mid (q, h, l, r) \in \delta \text{ for some } l \in L \} \]

The underlying L-automaton of a system \( A \) is \( A^L = (Q_L, L, \delta^L_L, \lambda(q_0)) \); similarly, the underlying H-automaton of \( A \) is \( A^H = (Q_H, H, \delta^H_H, \chi(q_0)) \). The underlying L-automaton contains Larry’s possible observations on the system \( A \), whereas the underlying H-automaton gives Harry’s possible observations.

The L-projection of an \( A \)-run \( \rho = (q_i \xrightarrow{h_i, l_i} q_{i+1})_{1 \leq i \leq n} \) is the run \( \rho^L = (\lambda(q_i-1) \xrightarrow{h_i} \lambda(q_i))_{1 \leq i \leq n} \).

This run is what Larry sees when \( \rho \) happens in the system \( A \). Similarly, the H-projection of \( \rho \) is \( \rho^H = (\chi(q_i-1) \xrightarrow{h_i} \chi(q_i))_{1 \leq i \leq n} \).

**Definition 2.** An \( \infty \)-strategy (or strategy of depth \( \infty \)) for player \( X \in \{H, L\} \) is a mapping \( s : (Q_X)^\infty \to X \). An \( n \)-strategy (or strategy of depth \( n \)) for player \( X \) is a mapping \( s : (Q_X)^{\leq n-1} \to X \), where \( n \in \mathbb{N} \).

Note that an \( n \)-strategy is defined on sequences of high-level states of length \( \leq n - 1 \). Also, \( s(\varepsilon) \) gives the decision that player \( X \), according to strategy \( s \), inputs into the system at its initial state.

An \( \infty \)-strategy for player \( H \) encodes the choices that Harry makes function of his observations of the system states. Note that we assume that Harry does not have access to the whole system state when he makes his decisions. The same observations hold for player \( L \).

The set of \( \infty \)-strategies and the set of \( n \)-strategies for player \( X \) are denoted respectively, \( \text{Str}^X_\infty \) and \( \text{Str}^X_n \) (\( X \) is here \( H \) or \( L \)). In the rest of this paper, unless explicitly stated, the term \( n \)-strategy refers to an \( n \)-strategy for \( H \).

If \( n \leq m \), we say that an \( n \)-strategy \( s \) is a prefix of an \( m \)-strategy \( s' \) iff \( s'(w) = s(w) \) for all \( w \in (Q_X)^{\leq n-1} \). By generalizing, we say that \( s \) is a prefix of an \( \infty \)-strategy \( s'' \) if the same property holds. In fact, we will denote \( n \leq \infty \) for any \( n \in \mathbb{N} \cup \{\infty\} \). We also denote \( s \preceq s' \) if strategy \( s \) is a prefix of strategy \( s' \).

For the sake of completion, we define here also strategies for the system: a strategy for the automaton \( A \) is a mapping \( s : Q \times H \times L \to Q \) satisfying the following consistency requirement:

\[ \forall q \in Q, h \in H, l \in L, s(q, h, l) \in \delta(q, h, l) \]

In other words, a strategy for the system is a means for the system to “solve” nondeterminism. We may define also \( n \)-strategies for \( A \) following the same pattern of Definition 2.

**Definition 3.** Given an \( m \)-strategy \( s \ (m \in \mathbb{N} \cup \{\infty\}) \) and a run \( \rho = (q_i \xrightarrow{h_i, l_i} q_{i+1})_{1 \leq i \leq n} \) with \( n \leq m - 1 \), we say that \( s \) is compatible with \( \rho \) if the following condition is satisfied:

\[ s(\varepsilon) = h_1 \text{ and } \forall 1 \leq i < n, s(\chi(q_1) \ldots \chi(q_i)) = h_{i+1} \]

The set of behaviors observable by \( L \) when \( H \) acts following strategy \( s \) is

\[ \text{Obs}_L(s) = \{ \rho^L \mid \rho \text{ compatible with } s \} \]
In other words, \( s \) is compatible with \( \rho \) if \( \rho^H \) contains the sequence of “decisions” that Harry makes when acting like \( s \), in accordance with the part of the current state that he may observe.

Figure 1 gives an example of a game automaton. In this figure we denote states as tuples \((q, r)\), meaning that \( \chi(q, r) = q \) and \( \lambda(q, r) = r \). Also the loops in the “terminal” states are labeled with any tuple of high and low actions.

![Fig. 1. A system](image)

The following two mappings are strategies in the system in Figure 1:

\[
\begin{align*}
\sigma_1(\varepsilon) &= h_1 & \sigma_1(q_1) &= h_1 & \sigma_1(q_1, q_2) &= h_1 & \sigma_1(w) &= h_1 \text{ otherwise} \\
\sigma_2(\varepsilon) &= h_1 & \sigma_2(q_1) &= h_2 & \sigma_2(q_1, q_3) &= h_2 & \sigma_2(w) &= h_1 \text{ otherwise}
\end{align*}
\]

The run \((q_0, r_0) \xrightarrow{h_1, l} (q_1, r_1)\) is compatible with both strategies, whereas the run \((q_0, r_0) \xrightarrow{h_1, l} (q_1, r_1) \xrightarrow{h_2, l} (q_3, r_1)\) is compatible only with strategy \( \sigma_2 \). It is easy to see that \( \text{Obs}(\sigma_2) \setminus \text{Obs}(\sigma_1) = \{r_0 \xrightarrow{l} r_1 \xrightarrow{l} r_1\} \).

The main notion of this section is the following, which generalizes the Nondeducibility on Strategies of Wittbold & Johnson, [WJ90] and is closely related to the approach of Lowe in [Low04]:

**Definition 4.** The covert channel capacity allowed by the system \( A \) is

\[
K_A = \text{card}(B_A) - 1 \quad \text{where} \quad B_A = \{\text{Obs}_L(s) \mid s \in \text{Str}_H^{\infty}\}
\]

A system \( A \) has zero covert channel capacity (ZCCC) if \( K_A = 0 \).

Hence, in a system with covert channel capacity 2, a “Trojan Harry” may be able to send one bit of information to Larry, according to one of the two classes of \( \infty \)-strategies that it chooses.

On the other hand, the above definition does not make any difference between strategies that deadlock the system and strategies that do not do this. Take, for example, the systems in Figure 2, which both have \( K_A = 4 \).

![Fig. 2. Two systems](image)

Though both systems have the same covert-channel capacity, we may still consider system (b) better than system (a): consider only strategies that, for any system choice, do not block the system. Then in (a) Harry has two such classes of strategies, whereas in (b) Harry has only one, namely:

\[
\begin{align*}
\sigma(\varepsilon) &= h_1 & \sigma(q_1) &= h_1 & \sigma(q_2) &= h_2 & \sigma(z) &= \text{arbitrary, for other values of} \ z
\end{align*}
\]
Proposition 1.

Proof. The set of admissible \( m \)-strategies for \( n \) is compatible with \( m \) that have to be done when the system does not produce the run \((q_2, r_1)\) or to \((q_3, r_1)\), whereas at \( b \) it is the system’s decision to go to \((q_1, r_1)\) or \((q_2, r_1)\). We think that this difference is quite important, since we consider the system as a reliable component, only Harry is unreliable.

The solution is to count only the strategies that may never put the system in a deadlocked state – strategies that are not detectable by some “deadlock-based intrusion-detection system”. We will therefore call:

**Definition 5.** An \( n \)-strategy \( s \) for \( X \in \{ H, L \} \) is admissible \(( n \in \mathbb{N} \cup \{ \infty \} )\) if every run \( \rho \) of length \( m \leq n \) which is compatible with \( s \) is a prefix of a run \( \rho' \) of length \( n \) which is compatible with \( s \) too. The set of admissible \( n \)-strategies for \( X \) \(( n \in \mathbb{N} \cup \{ \infty \} )\) and \( X \in \{ H, L \} \) is denoted \( \text{Adm}_H^n \).

An interesting property is stated in the following:

**Proposition 1.** For any infinite \( A \)-run \( \rho \) there exists some admissible \( \infty \)-strategy \( s \in \text{Adm}_H^\infty \) which is compatible with \( \rho \).

**Proof.** The idea is that the \( H \)-projection of the states in the run, together with the \( H \)-actions in it, give the “variant” of the strategy that will be compatible with \( \rho \). All that remains to build are the choices that have to be done when the system does not produce the run \( \rho \). And this is easily done by relying on the “liveness” assumption 1 from Definition 1.

Suppose \( \rho = \left( q_{n-1} \xrightarrow{h_{n-1}L_{n-1}} q_n \right)_{n \geq 1} \). Then we put \( \forall n \geq 1, s(\chi(q_1) \ldots \chi(q_{n-1})) = h_n \).

Now, we define the rest of the strategy \( s \). Let \( k \geq 0 \) and assume we have defined \( s(r_1 \ldots r_i) = h_{i+1}' \) for any \( 0 \leq i \leq k \), and for any \( r_1, \ldots, r_k \in Q_H \). Also, let \( r \in Q_H \). We have two cases:

- Suppose \( q_0 \xrightarrow{h_0' l_0} t_1 \xrightarrow{h_2' l_2} \ldots \xrightarrow{h_{k'} l_{k'}} t_k \) is a run in \( A \), with \( \chi(t_i) = r_i \). Two sub-cases may occur:
  - \((t_k, h_{k+1}', l_{k+1}, q) \in \delta\) for some \( l \in L \) and \( q \in Q \) with \( \chi(q) = r \); then by liveness we must also have \((q, h', l', q') \in \delta\) for some appropriate \( h', l', q' \). We then define \( s(r_1 \ldots r_k r) = h' \).
  - otherwise, we put an arbitrary value for \( s(r_1 \ldots r_k r) \).

- otherwise, we put an arbitrary value for \( s(r_1 \ldots r_k r) \).

Clearly, the \( \infty \)-strategy constructed here is an admissible strategy.

**Definition 6.** The admissible covert channel capacity allowed by the system \( A \) is

\[
K_{aA} = \text{card}(BA_A) - 1 \quad \text{where} \quad BA_A = \left\{ \text{Obs}_L(s) \mid s \in \text{Adm}_H^\infty \right\}
\]

A system \( A \) has zero admissible covert channel capacity (ZACCC) if \( K_{aA} = 0 \).

With this new definition, the system \( a \) in Figure 2 has \( K_{aS} = 1 \), whereas the system \( b \) has \( K_{aS} = 0 \). It is straightforward to observe that covert channel capacity in the sense of Definition 4 can be also recovered by inserting an error state giving a special, new observation \( \bot \) and transforming the transition relation such that the undefined transitions be replaced with transitions leading to the error state. In other words, if we wish admitting deadlocking strategies, then we might be interested in making deadlocks visible.

Before ending this section, we prove a result that will show its use in the next section:

**Proposition 2.** Given an admissible \( n \)-strategy \( s \), \( n \in \mathbb{N} \), there exists an admissible \( \infty \)-strategy \( s' \) with \( s \preceq s' \).
Definition 7. Given a run $\theta$ of length $n$ in the underlying $L$-automaton $A_L$, Larry's knowledge after observing $\theta$ is the set

$$\text{knl}(\theta) = \{ s \in \text{Adm}^{n}_{H} \mid \exists \rho \in \text{Runs}(A) \text{ s.t. } \theta = \rho|_{L} \text{ and } s \text{ is compatible with } \rho \}$$

A game automaton $A$ has no deducible information flow if for any two $A_L$-runs $\theta_1, \theta_2 \in \text{Runs}(A_L)$ with $\theta_1 \preceq \theta_2$,

$$\text{knl}(\theta_1) \preceq \text{knl}(\theta_2)$$

Here we have extended the relation $\preceq$ to sets of strategies in the obvious manner, i.e. $A \preceq B$ iff $\forall s \in A, \exists s' \in B$ with $s \preceq s'$.

Theorem 1. An automaton $A$ has zero admissible covert channel capacity if and only if it has no deducible information flow.

Let us associate first, to each $A_L$-run $\theta$, the following set of admissible $\infty$-strategies:

$$\text{genknl}(\theta) = \{ s \in \text{Adm}^{\infty}_{H} \mid \exists \rho \in \text{Runs}(A) \text{ s.t. } \theta = \rho|_{L} \text{ and } s \text{ is compatible with } \rho \}$$

We will actually prove the following property:

Proposition 3. A has no deducible information flow if and only if for any two $A_L$-runs $\theta_1, \theta_2$ with $\theta_1 \preceq \theta_2$, we have that $\text{genknl}(\theta_1) = \text{genknl}(\theta_2)$.

It is easy to see that, once this lemma is proved, Theorem 1 is proved too.

Proof (Of the Proposition 3). Suppose there exist two $A_L$-runs $\theta_1, \theta_2$ with $\theta_1 \preceq \theta_2$ and $\text{genknl}(\theta_1) \neq \text{genknl}(\theta_2)$. Without loss of generality, we may assume that $\theta_2 = \theta_1 \xrightarrow{l} r$, for some $l \in L$ and $r \in Q_L$ — that is, $\theta_1$ is an immediate prefix of $\theta_2$.

It is easy to see that $s \in \text{genknl}(\theta_2)$ implies that $s \in \text{genknl}(\theta_1)$, since for each $A$-run $\rho$ of length $\ell(\theta_2)$ that is compatible with $s$, if we take its immediate prefix $\rho'$, we have that $\rho'|_{L} = \theta_1$ and $\rho'$
is compatible with \( s \). So the assumption \( \text{genknl}(\theta_1) \neq \text{genknl}(\theta_2) \) implies that we must have some \( \infty \)-strategy \( s \in \text{genknl}(\theta_1) \setminus \text{genknl}(\theta_2) \).

Denote \( s' \) the restriction of \( s \) to \((Q_H)^{\ell\text{en}(\theta_1)-1}\). Then clearly \( s' \in \text{kn}(\theta_1) \). On the other hand, since \( s \notin \text{genknl}(\theta_2) \), \( s' \) cannot be the prefix of a \((\ell\text{en}(\theta_2)-1)\)-strategy in \( \text{kn}(\theta_2) \), fact which proves that \( \text{kn}(\theta_1) \notin \text{kn}(\theta_2) \) and hence the direct implication is proved.

For the reverse implication, suppose that \( \text{kn}(\theta_1) \neq \text{kn}(\theta_2) \) and denote \( n = \ell\text{en}(\theta_1) \). We will assume again, without loss of generality, that \( \theta_2 = \theta_1 \overset{L}{\rightarrow} r \), for some \( l \in L \) and \( r \in Q_L \) – that is, \( \theta_1 \) is an immediate prefix of \( \theta_2 \).

Take then \( s \in \text{kn}(\theta_1) \) an \( n \)-strategy for which there exists no \((n+1)\)-strategy \( s' \in \text{kn}(\theta_2) \) with \( s \preceq s' \). In particular, we get that for any run \( \rho \) of length \( n+1 \) for which \( \rho_{|L} = \theta_2 \), \( s \) can not be the prefix of an \((n+1)\)-strategy compatible with \( \rho \).

On the other hand, with the aid of Proposition 2 we may build an admissible \( \infty \)-strategy \( s'' \) such that \( s \preceq s'' \). It then follows that \( s'' \notin \text{genknl}(\theta_1) \).

But since \( s \) can not be the prefix of an \((n+1)\)-strategy compatible with \( \rho \) of length \( n+1 \) for which \( \rho_{|L} = \theta_2 \), we obtain that \( s'' \notin \text{genknl}(\theta_2) \).

\[ \square \]

**Remark 1.** Note that, if \( \theta \) is an \( A_L \)-run then \( \text{genknl}(\theta) \) is nonempty. This property follows from the fact that any run \( \rho \in \text{Runs}(\mathcal{A}) \) is compatible with an admissible \( \infty \)-strategy, as stated in Proposition 1.

### 4 Decidability

In this section we will give an algorithm for checking whether a finite-state game automaton has the ZACCC property. In other words, we will show that the following problem is decidable:

**Problem 1.** The zero admissible covert channel capacity problem (ZACCC problem): Given a finite-state system \( \mathcal{A} \), does it have the ZACCC property?

**Theorem 2.** The ZACCC problem is decidable.

**Proof.** We build a nondeterministic finite automaton whose states encode pairs of strategies in the initial system. We then check whether there exists a reachable state in which one of the strategies accepts a low-level event that the other strategy forbids.

Formally, suppose that the initial finite-state system is \( \mathcal{A} = (Q, Q_H, Q_L, H_L, L, \delta, q_0, \chi, \lambda) \). Then we build the nondeterministic finite automaton \( \overline{\mathcal{A}} = (\mathcal{P}(Q) \times \mathcal{P}(Q), L, \overline{\delta}, (\{q_0\}, \{q_0\})) \), where \( \overline{\delta} \) contains a tuple \( ((S_1, S_2), l, (S'_1, S'_2)) \) if and only if for both \( i = 1, 2 \), if we enumerate the states of \( S_i = \{q^i_1, \ldots, q^i_{n_i}\} \) then there exist \( h^i_1, \ldots, h^i_{n_i} \in H \) such that

(a) \( S'_i = \bigcup_{1 \leq j \leq n_i} \delta(q^i_j, h^i_j, l) \)

(b) For all \( 1 \leq j_1, j_2 \leq n_i \), if \( \chi(q^i_{j_1}) = \chi(q^i_{j_2}) \) then \( h^i_{j_1} = h^i_{j_2} \).

In other words, each component in a reachable state \( (S_1, S_2) \) in \( \overline{\mathcal{A}} \) contains all the states that belong to a run that is compatible with some given strategy.

Given a run \( \theta = ((S_1^{i-1}, S_2^{i-1}) \overset{L}{\rightarrow} (S_1^i, S_2^i))_{1 \leq i \leq n} \) in \( \overline{\mathcal{A}} \), we say that \( \theta \) 1-encumbers the following set of runs:

\[
\text{Enc}_1(\theta) = \{ \rho = (q_{i-1} \overset{h_{i,i}}{\rightarrow} q_i)_{1 \leq i \leq n} \in \text{Runs}(\mathcal{A}) \mid \forall 1 \leq i \leq n, q_i \in S^i_1, h_i \in H \}
\]
Similarly, we say that $\theta$ 2-encompasses the following set of runs:

$$\text{Enc}_2(\theta) = \{ \rho = (q_{i-1} \xrightarrow[h_i, l_i]{} q_i)_{1 \leq i \leq n} \in \text{Runs}(A) \mid \forall 1 \leq i \leq n, q_i \in S^i_2, h_i \in H \}$$

We will prove the following

**Claim. (§)**

1. Given a run $\theta = ((S^i_1, s_{i-1}^i) \xrightarrow{l_i} (S^i_2, s_{i+1}^i))_{1 \leq i \leq n}$ in $\overline{A}$, there exists an $n$-strategy $s_1$ for $H$ which is compatible with all the runs in $\text{Enc}_1(\theta)$; we will call $s_1$ 1-associated with $\theta$. A similar property holds for $\text{Enc}_2(\theta)$.

2. For each pair of $n$-strategies $s_1, s_2$ there exists a run $\theta = ((S^i_1, s_{i-1}^i) \xrightarrow{l_i} (S^i_2, s_{i+1}^i))_{1 \leq i \leq n}$ in $\overline{A}$ which is 1-associated with $s_1$ and 2-associated with $s_2$.

**Proof (Of the Claim (§)).** We prove the first part of this claim by induction on $n = \ell(\theta)$; suppose the claim is proved for all runs of length $\leq n - 1$ and consider a run $\theta$ of length $n$. By induction, we have an $(n - 1)$-strategy $s_1$ associated with the immediate prefix of $\theta$.

To construct an $n$-strategy $s_2$ with $s_1 \leq s_2$ and which is associated with $\theta$, we clearly must put $s_2(z) = s_1(z)$ for all $z \in (Q_H)^{\leq n-2}$. Consider then an enumeration of $S^{n-1}_1 = \{q_1, \ldots, q_{n_1}\}$. By construction of $\overline{A}$, we must have $n_1$ symbols $h_{1}, \ldots, h_{n_1} \in H$ such that $S^j_1 = \bigcup_{1 \leq j \leq n} \delta(q_j, h_j, l_n)$ and with the additional property (b) above.

Then, for each $A$-run $\rho = (r_{i-1} \xrightarrow[l_i]{} r_i)_{1 \leq i \leq n-1}$ associated with $s_1$ for which $\chi(r_{n-1}) = \chi(q_j)$ for some $1 \leq j \leq n_1$, we put

$$s_2(\chi(r_1) \ldots \chi(r_{n-1})) = h_j$$

(2)

Let us prove that this is indeed a functional mapping, that is, that if we have two runs ending in states with the same $H$-projections, then the two respective definitions of the type 2 are not contradictory.

To this end, suppose we have two runs $\rho_m = (r^m_{i-1} \xrightarrow[l_i]{} r^m_i)_{1 \leq i \leq n-1} \in \text{Enc}_1(\theta)$ compatible with $s_1$ ($m = 1, 2$) and with $\chi(r^1_{n-1}) = \chi(r^2_{n-1})$. Since $r^1_{n-1}, r^2_{n-1} \in S^{n-1}_1$, we have $r^1_{n-1} = q_{j_1}$ and $r^2_{n-1} = q_{j_2}$. But the construction of $\overline{A}$ implies that $h_{j_1} = h_{j_2}$, hence the choice of $h_j$ in the Identity 2 is unique.

Hence, the first property in the Claim (§) is proved.

In order to prove the second property, for each $n$-strategy $s$ we construct the set of $A$-runs compatible with $s$ which have the same $L$-inputs – that is, given $\overline{t} = (l_1, \ldots, l_n) \in L^n$, define

$$\text{Cmpt}(s, \overline{t}) = \{ \rho = (q_{i-1} \xrightarrow[l_i]{} q_i)_{1 \leq i \leq n} \mid \rho \text{ compatible with } s \}$$

Then, given two $n$-strategies $s_1$ and $s_2$ and an $n$-tuple of $L$-events $\overline{l} = (l_1, \ldots, l_n) \in L^n$, we define the $\overline{A}$-run $\theta = ((S^i_1, s_{i-1}^i) \xrightarrow[l_i]{} (S^i_2, s_{i+1}^i))_{1 \leq i \leq n}$ in which

$$S^j_2 = \{ q \mid \exists \rho \in \text{Cmpt}(s_j, \overline{l}), \rho[1 \ldots i] \text{ ends in } q \}$$

and $S^0_2 = \{ q_0 \}$. It is easy to see then that

$$\text{Cmpt}(s_1, \overline{t}) = \text{Enc}_1(\theta) \quad \text{Cmpt}(s_2, \overline{t}) = \text{Enc}_2(\theta)$$

That is, $s_1$ is 1-associated with $\theta$ and $s_2$ is 2-associated with $\theta$. This ends the proof of the Claim (§).

**Proof (Of Theorem 2, continued).** We will prove the following property:
A has the ZACCC property if and only if, in \( \overline{A} \), there exists an accessible macro-state \((S_1, S_2)\) with \( \lambda(S_1) \neq \lambda(S_2) \).

Suppose first there exists \((S_1, S_2)\) accessible in \( \overline{A} \) with \( \lambda(S_1) \neq \lambda(S_2) \). Let then \( \theta \) be a run in \( \overline{A} \) which ends in \((S_1, S_2)\). Observe then that \( \text{Enc}_1(\theta)[L] \neq \text{Enc}_2(\theta)[L] \). But, by the Claim (*), there exist two strategies \( s_1 \) and \( s_2 \) of depth \( \ell(\theta) \) such that \( \text{Cmpt}(s_1, \bar{t}) = \text{Enc}_1(\theta) \) and \( \text{Cmpt}(s_1, \bar{t}) = \text{Enc}_2(\theta) \).

Both \( s_1 \) and \( s_2 \) being admissible, by means of Proposition 2 we may build two \( \infty \)-strategies \( \bar{s}_1 \) and \( \bar{s}_2 \) with \( s_j \leq \bar{s}_j \), which, in fact, mean that \( \text{Cmpt}(s_j, \bar{t}) = \text{Cmpt}(\bar{s}_j, \bar{t}) \) \((j = 1, 2)\). It then follows that \( \text{Obs}(\bar{s}_1) \neq \text{Obs}(\bar{s}_2) \), which means that \( A \) does not have the ZACCC property.

For the reverse proof, take \( s_1 \) and \( s_2 \) two \( \infty \)-strategies with \( \text{Obs}(s_1) \neq \text{Obs}(s_2) \). Without loss of generality, we may assume that there exists \( \rho \in \text{Runs}(A) \) with \( \rho|_L \in \text{Obs}(s_1) \setminus \text{Obs}(s_2) \).

Consider \( \rho' \) the longest prefix of \( \rho \) for which \( \rho'|_L \in \text{Obs}(s_1) \cap \text{Obs}(s_2) \), and take \( \rho'' = \rho' \xrightarrow{h,l} q \leq \rho \).

Hence \( \rho'' \) is compatible with \( s_1 \) but not with \( s_2 \). Denote \( n = \ell(\rho') \) and \( \bar{t} = l_1 \ldots l_n \) the sequence of \( L \)-inputs in \( \rho' \).

Now we construct, according to Claim (*), the \( \overline{A} \)-run \( \theta \) of length \( n + 1 \) which is \( 1 \)-associated with \( s_1 \) and \( 2 \)-associated with \( s_2 \) and has its transitions labeled with \( l_1, \ldots, l_n, l \). Suppose its last state transition is \((S_1, S_2) \xrightarrow{l} (S'_1, S'_2)\).

Observe first that, since \( \rho'' = \rho' \xrightarrow{l} q \) is compatible with \( s_1 \), we must have \( q \in S'_1 \). But, if there existed \( q' \in S_2 \) with \( \lambda(q') = \lambda(q) \) then there would also exist \( h' \in H \) with \( \rho'' \xrightarrow{h',l} q' \) compatible with \( s_2 \), which would imply that \( \rho''|_L \in \text{Obs}(s_2) \), contrary to our assumption.

This, together with the fact that reachability is decidable in finite automata, completes the proof of Theorem 2. \( \square \)

5 Relationship to some existing notions of information flow

In this section we will investigate the relationship with some existing trace models of information flow, translated in our framework. The reason of this translation is that existing models, apart Wittbold & Johnson, make use of asynchronous models in which Harry and Larry play with the system without being forced to react at each moment.

5.1 Relationship with Generalized Noninterference

Throughout this subsection, we will suppose that the system \( A \) is \( H \)-input total, that is

\[
\forall q \in Q, \forall h \in H \exists l \in L \text{such that } \delta(q, h, l) \neq \emptyset
\]

Generalized Noninterference (GNI, [McC87]) is a nondeterministic generalization of the classical notion of Noninterference [GM82]. It states that, given two traces, if we “recombine” the high-level input from one trace with the low-level behavior of another one, then the result may be “corrected” into a trace of the system by modifying only high-level outputs.

Originally, GNI utilizes interleaving as “recombination”. In our synchronous setting however, interleaving is not a good option since it introduces easily information flow through timing. We have chosen to give a “synchronous” variant of GNI – a choice which is also consistent with our approach.

**Definition 8.** A system \( A \) satisfies Synchronous Generalized Noninterference (SGNI) if it is \( H \)-input total and it satisfies the following property:
For any two runs $\rho, \rho' \in \text{Runs}(A)$, with $\rho = (q_i, \ell_i, q_{i+1})_{1 \leq i \leq n}$, there exists a run $\rho'' = (r_{i-1} \xrightarrow{h_i, \ell} r_i)$ with $\rho''|_L = \rho'|_L$.

Note that the sequence of $H$-inputs in both $\rho$ and $\rho''$ is the same. Figure 3 shows two examples of systems which satisfy SGNI.

![Fig. 3. Two examples of systems satisfying SGNI but not satisfying ZACCC.](image)

Note however that both systems in Figure 3 do not have the ZACCC property. To see this for the system at (a), consider the following strategies:

\[
\begin{align*}
    s_1(\varepsilon) &= h_1 & s_1(q_1) &= h_1 & s_1(q_2) &= h_1 & s_1(z) &= \text{arbitrary, otherwise} \\
    s_2(\varepsilon) &= h_1 & s_2(q_1) &= h_1 & s_2(q_2) &= h_2 & s_2(z) &= \text{arbitrary, otherwise}
\end{align*}
\]

We then have that $\text{Obs}(s_1) \neq \text{Obs}(s_2)$, since the $L$-run $r_0 \xrightarrow{\ell} r_1 \xrightarrow{\ell_2} r_2$ is in $\text{Obs}(s_2) \setminus \text{Obs}(s_1)$.

This result can be interpreted as follows: a Trojan Horse can be inserted at high level, and if it chooses a strategy of the type $s_2$ Larry has a chance of observing $r_2$, whereas if the Trojan chooses another strategy, Larry doesn’t have this chance. Of course, a more proper treatment of the word “choice” would require a probabilistic model of the system, which is out of the scope of this paper.

In the absence of a probabilistic setting, we may consider that the system at (a) does not allow Larry to have a chance to detect that his Trojan has chosen strategy $s_1$, since the runs compatible with $s_2$ are included in the runs compatible with strategy $s_1$. Figure 3 (b) shows a system in which not only a Trojan Horse has the possibility to choose between two strategies, but also Larry has a chance for observing a difference in each case – the two distinctly observable strategies are:

\[
\begin{align*}
    s_1(\varepsilon) &= h_1 & s_1(q_1) &= h_2 & s_1(q_2) &= h_2 & s_1(q_3) &= h_3 \\
    s_2(\varepsilon) &= h_1 & s_2(q_1) &= h_2 & s_2(q_2) &= h_3 & s_2(q_3) &= h_3
\end{align*}
\]

We may see that the $L$-run $r_0 \xrightarrow{\ell} r_1 \xrightarrow{\ell} r_3$ does not belong to $\text{Obs}(s_1)$, whereas the $L$-run $r_0 \xrightarrow{\ell} r_1 \xrightarrow{\ell} r_2$ does not belong to $\text{Obs}(s_2)$. So, if the Trojan Horse chooses $s_1$, there is a chance for Larry to observe this this, and similarly for a choice for $s_2$.

We may also prove that, even in the case of $H$-input total systems, ZCCC does not imply SGNI. The example that shows this is in Figure 4.
In Figure 4, any strategy $s$ is compatible with any run. However if we choose the sequence of $H$-inputs $h = h_1, h_1$ and the $L$-run $r^L = r_0 \xrightarrow{l_1} r_1 \xrightarrow{l_1} r_2$, we see that there is no run $\rho$ which has $h$ as its sequence of $H$-inputs and with $\rho|_L = r^L$.

We end this subsection with a strategy-based variant of Generalized Noninterference that makes reference to strategies:

**Definition 9.** A system $A$ satisfies **Noninterference on Strategies** (NoS) if it satisfies the following property:

For any two runs $\rho, \rho' \in \text{Runs}(A)$, and for each $\omega$-strategy $s$ which is compatible with $\rho$, there exists a run $\rho''$ which is compatible with $s$ and with $\rho''|_L = \rho'|_L$.

**Proposition 4.** A system satisfies NoS if and only if it has the ZACCC property.

*Proof.* Suppose the system $A$ satisfies NoS, and take two $\omega$-strategies $s_1, s_2$. Note first that in an input-total system, any strategy is admissible. Consider $\rho$ a run compatible with $s_1$ and $\rho'$ a run compatible with $s_2$. By NoS, we may find a run $\rho''$ which is compatible with $s_2$ and with $\rho''|_L = \rho'|_L$. This means that for any $\text{Obs}(s_1) \subseteq \text{Obs}(s_2)$; the reverse can be proved similarly, hence $A$ satisfies ZACCC.

Suppose now that $A$ satisfies ZACCC and take two runs $\rho, \rho'$ and an $\omega$-strategy $s$ compatible with $\rho$. By liveness, $\rho'$ can be extended to an infinite run $\overline{\rho'}$ and therefore there exists an $\omega$-strategy $s'$ which is compatible with $\overline{\rho'}$. But ZACCC implies that $\overline{\rho'}|_L \in \text{Obs}(s') = \text{Obs}(s)$, which gives exactly the NoS property for $s$, that is, there exists a run $\rho''$ with $\rho''|_L = \rho'|_L$ and compatible with $s$. \qed

### 5.2 Comparing strategy-based semantics and bisimulation-based semantics

In this section we compare our model of information flow with the bisimulation-based model of [FG95,FG00]. Recall that, in their framework, a system satisfies **Bisimulation-based Nondeducibility on Compositions** if, for any two high-level processes, if we compose the system with each high-level process and remove any non-synchronizing high-level activity, we get two bisimilar processes.

In our setting, the rôle of high-level processes is taken by *strategies*. Hence, we first need to specify in a form of an automaton the composition between a strategy and a system. Then we define a bisimulation relation which takes into account only low-level activity. Note that in our framework high-level activity is always synchronized with low-level activity, hence there is no need for “removing non-synchronized high-level activity”.

![Fig. 4](image-url)
Given an \( \infty \)-strategy for \( H \) \( s \in \text{Str}_H^\infty \), the \textit{s-governed system} \( \mathcal{A}(s) \) is the system \( \mathcal{A}(s) = (\mathcal{R}, Q_H, Q_L, H, L, \delta, q_0, \chi, \lambda) \) where
\[
\mathcal{R} = \{(q, z) \mid z \in (Q_H)^*, z = z'r, r \in Q_H, \chi(q) = r\}
\]
\[
\delta = \{((q, z), h, l, (q', z\chi(r))) \mid z \in (Q_H)^*, (q, h, l, r) \in \delta, s(z) = h\}
\]
\[
\tilde{\chi}((q, z)) = \chi(r) \text{ if } z = z'r \text{ for some } r \in Q_H
\]
\[
\tilde{\lambda}((q, z)) = \lambda(q)
\]
\( \mathcal{A}(s) \) is an automaton model for the composition between a system and a high-level strategy.

Given two systems \( \mathcal{A} = (Q, Q_H, Q_L, H, L, \delta, q_0, \chi, \lambda) \), and \( \mathcal{A} = (\overline{Q}, \overline{Q}_H, Q_L, \overline{H}, H, L, \overline{\delta}, \overline{q}_0, \overline{\chi}, \overline{\lambda}) \), a \textit{bisimulation} between \( \mathcal{A} \) and \( \overline{\mathcal{A}} \) is a relation \( \equiv \subseteq Q \times \overline{Q} \) having the following properties:

1. For each \( q \equiv \overline{q} \) and each \((h, l) \in H \times L\), if \( q \xrightarrow{h,l} r \) then there exists \( \overline{r} \in \overline{H} \) and \( \overline{q} \in \overline{Q} \) such that \( \overline{q} \xrightarrow{h',l'} \overline{r} \) and \( \overline{\chi}(\overline{r}) = \lambda(r) \).
2. For each \( q \equiv \overline{q} \) and each \((\overline{h}, l) \in \overline{H} \times L\), if \( \overline{q} \xrightarrow{\overline{h},l} \overline{r} \) then there exists \( h \in H \) and \( r \in Q_H \) such that \( q \xrightarrow{h,l} r \) and \( \overline{\chi}(\overline{r}) = \lambda(r) \).

Note that the two automata must share both the \( L \)-input and the \( L \)-output sets.

A system has the \textit{bisimulation-based nondeducibility on composition} (BNDC, see [FG00]) property if for any two \( \infty \)-strategy \( s_1, s_2 \in \text{Str}_H^\infty \), \( \mathcal{A}(s_1) \equiv \mathcal{A}(s_2) \).

Note that we do not need to “remove high level events” as in [FG00], since our bisimulation definition speaks only of low-level events.

Consider the system in Figure 5

![Figure 5. An example of a system which is not BNDC but has the ZCCC.](image)

This system does not satisfy the BNDC property, since if we take some strategy \( s_1 \) for which \( s_1(\varepsilon) = h_1 \) and another strategy \( s_2 \) for which \( s_2(\varepsilon) = h_2 \), we get \( \mathcal{A}(s_1) \neq \mathcal{A}(s_2) \). However it is easy to see that for any strategy \( s \in \text{Str}_H^\infty \), \( \text{Obs}(s) \) is the same.

In this example the states \((q_2, r_1)\) and \((q_9, r_1)\) cannot be bisimilar, since the set of system choices in these two states are not the same. But in our framework information flow is not induced by system choices but by \textit{Harry’s choices}! This example suggests that, in our setting, “strategy-based semantics” is better than “bisimulation-based semantics”.

14
6 Constructing a system controller for avoiding covert channels

This section is dedicated to the following problem:

Problem 2 (The infoflow controller synthesis problem). Given a system $A = (Q, Q_H, Q_L, H, L, \delta, q_0, \chi, \lambda)$ that does not satisfy the ZACCC property, can we reduce it, i.e. construct another system, $\tilde{A} = (\tilde{Q}, Q_H, Q_L, H, L, \tilde{\delta}, q_0, \chi, \lambda)$, with $\tilde{Q} \subseteq Q$ and $\tilde{\delta} \subseteq \delta$ and which satisfies this property?

The solution to this problem stems from the possibility for the system to “resolve nondeterminism” – or, in other words, to use a strategy for the system, acting like a controller which does not allow information leaks.

It is trivial that any system can be reduced to a system satisfying the ZACCC property by simply choosing any infinite run $\rho$ and allowing only this run to be executed in the system. This is clearly not a satisfactory solution in general.

On the other hand, note that there is no “maximal” automaton $\tilde{A}$ (w.r.t. state inclusion) which solves the infoflow controller synthesis problem. For example, for the automaton in Figure 2 (a), there are two sub-automata which solve this problem: the one using states $\tilde{Q} = \{(q_0, r_0), (q_1, r_1), (q_2, r_1), (q_4, r_1)\}$ and the one using $Q_H = \{(q_0, r_0), (q_1, r_1), (q_3, r_1), (q_5, r_2)\}$.

We will use our construction of the automaton $\tilde{A}$ for deciding whether a system satisfies ZACCC to provide an algorithm for solving the infoleak controller synthesis problem.

The idea is the following: suppose we have in $\tilde{A}$ a macro-state $(S_1, S_2) \in \mathcal{P}(Q) \times \mathcal{P}(Q)$ starting from which we may reach a state $(S'_1, S'_2)$ with with $\lambda(S'_1) \neq \lambda(S'_2)$. We pick some state $q \in S_1$ for which there exist $h_1 \neq h_2 \in H$, $l \in L$ and $r_1, r_2 \in Q$ with $(q, h_1, l, r_1), (q, h_2, l, r_2) \in \delta$. We then restrict $\delta$ to a new relation $\delta' = \delta \setminus \{(q, h_1, l, r_1)\}$

We rebuild the new automaton $A_1 = (Q, Q_H, Q_L, H, L, \delta', q_0, \chi, \lambda)$ and reiterate the whole procedure, until the resulting automaton satisfies ZACCC.

7 Conclusions

We have presented a synchronous model for the information flow problem. Our approach is based on three alternative definitions of information flow: the first defines covert channel capacity, i.e. the quantity of information that can be communicated from high to low; the second defines information flow by means of low-user’s deductions about high level strategies, by observing system behaviors. Finaly, the thirs notion is a “local correctability” condition that allows the design of a decision procedure for the information flow problem for a finite-state system.

A direction for further research is to investigate the relationship of our deduction model with deontic logic approaches [Cup93]. Another direction is to reformulate our results in the asynchronous model of [FG00] and to design a decision model for the BNDC property, [Mar98]. A third direction is to investigate in detail extensions of our results to mixed possibilistic/probabilistic models of information flow. A fourth direction is towards a thorough comparison between our framework and [Man00a].

References


