Nondeterministic noninterference and deducible information flow

Cătălin Dima* Constantin Enea† Radu Gramatovici‡

Abstract

We define two alternative notions related to the security of information flow. The first one is a weaker form of noninterference, which we call “nondeterministic noninterference”. It gives the view of the high-level user (Harry) on the requirements a system must satisfy in order for information about his activity not to be leaked to low-level. The second one is based on a semantic framework for defining the deductions that the low-level user makes by observing system behavior, and especially on the fact that a new deduction does not contradict a previous one. This second notion gives the low-level user (Larry’s) view on the characterization of the systems which do not allow him to gain information about high-level activity. We then show that the two views are equivalent.

The nondeterministic noninterference is based on a re-statement of the principle of noninterference. The main novelty in this notion is that it takes into consideration the possibility for the system to restrict Harry’s choices without impeding on information flow. This makes our notion strictly weaker than existing notions like the Generalized Noninterference, Forward Correctability, the Perfect Security Property and others.

In our framework, Harry’s choices are defined as strategies for Harry to interact with the system, in order to incorporate system nondeterminism that is inherent in possibilistic trace models. Hence, our restatement of noninterference is that “different high-level strategies should not lead to different observations by low-level users”. Alternatively, and equivalently, we propose Larry’s view of this principle, which would be that “high-level strategies should not be deduced by low-level users”. The equivalence of the two notions gives strong evidence that our framework might lay more solid foundations for a theory of information flow.

We also give an algorithm for deciding whether a finite-state system has no information flow in our setting.

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 trace-based possibilistic models [ZL97, Man00, McL94] which are able to detect a wide variety of types of information flow, including information flow through timing [FGM03]. However they are too strong as they label as information flow a simple program in which a high-level variable can be assigned constant values at different times. This is also one of the reasons for the proliferation of weaker and weaker trace-based properties, as newer models tried to allow more and more systems as nonleaking information.

This article started from the following simple observation: the main problem with the existing trace-based possibilistic definitions of information flow is that they start from the assumption that, at each moment, Harry, the high-level user, has unlimited ability to choose the command/instruction/input he wishes for interacting with the system. This assumption is unnecessary strong since it leads to labeling a trivial piece of code like

```plaintext
int high; read(high); write(2);
```

as leaking information, since Harry is forced to interact with the system – he cannot choose not to interact with it. Note that, even if we add some “nondeterminism”, allowing Harry not to interact with the system, the result would also be labeled – following Noninterference, the Perfect Security Property and Forward Correctability (see [ZL97, Man00]), as well as many other definitions of information flow – as allowing some kind of information leak because Harry cannot input a float into high!

---

*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
To properly define noninterference in possibilistic trace models, we need to take into account the nondeterminism that may hide system decisions. In a deterministic setting a choice for Harry could just be a sequence of high-level actions that he wishes to make apriori for interacting with the system. This is the basis for the trace definition of Generalized Noninterference and other related notions.

In a nondeterministic setting in which the system may also have the possibility to allow or to restrict, at times, Harry’s possibilities of interaction, a “choice” for Harry must incorporate the system restrictions at each moment. This brings us to the necessity to define strategies for Harry to interact with the system, as a generalization for the term “choice”. A strategy is a consistent set of decisions that Harry is promising to take at each moment, depending on the decisions of the system. The consistency requirement models exactly the necessity to consider system restrictions not as information flow but as common knowledge about the system, knowledge that Larry and Harry share.

This model is a sort of 1$^{1/2}$ player framework, in which Harry plays with the system, while Larry is an eavesdropper, being only able to observe system behaviors.

In this setting, we restate (nondeterministic) information flow as follows:

Different high-level strategies should not lead to different observations by low-level users.

This is, in our framework, Harry’s view on the security of information flow for the system.

The other approach taken in this paper is to define also an alternative, low-level view of the security of information flow in the system. The principles guiding this alternative approach are the following:

- Larry has a model of the system (apriori knowledge).
- Larry can partially observe system behavior through the time, and
- Using his apriori knowledge and his observations, Larry can infer new information about what happened in the system.

These principles are present, intuitively, in almost all papers on trace-based information flow, and are based on the hypothesis that Larry, the low-level user, knows the system specification but does not know the interaction of the system with its high-level environment. This assumption is common with [ZL97, Man00], which consider that, as soon as a trace $t$ occurs, and starting from the part of $t$ that can be observed at low level, Larry deduces that all the traces accepted by the system and which give the same observation as $t$ may have occurred. (This setting is also inspired from the so-called Dolev-Yao model of an attacker in security protocols [DY83].) However these principles have not yet been formalized into a self-contained theory that would lay the fundaments of information flow models. [BS97, OH03] investigate the possibility to define information flow logically, but [BS97] does not focus on sequential information flow, whereas the work of [OH03] relates only to noninterference or separability. The closest approach is the one in [CMS05], which defines the modifications in Larry’s belief as a means for quantifying probabilistic information flow.

The assumption is quite natural from several points of view, starting from the impossibility to design a model of information flow in which Larry has no knowledge about the system (how could he discover what’s behind the observable actions since he assumes from the beginning that anything may happen?). Secondly, Harry and Larry interact with the same system, which means that they have a set of “shared objects” (variables) via which information flows. The assumption allows a proper modeling of this shared objects. Finally, note that implicit information flow and information flow through timing are essentially due to this sharing of system variables (in the second case it is the “universal time” that is shared and not the system clock).

In our setting, besides having complete knowledge of the system specification, Larry is also endowed with unlimited deductive power. That is, Larry is able to deduce, before even the system starts running, where would Harry, the high-level user, be forced to react without any choice and where Harry can still make his own choices. This stronger assumption is appropriate for systems without encryption of information, but definitely not as a model for security protocols. Furthermore, we do not endow Larry with any capability of choosing his actions – i.e. Larry is an eavesdropper only.

The main target for Larry is then to find out the actual choices Harry would have done along his interaction with the system – at the moments where the system allows Harry to make choices. To this end, Larry makes the following type of deductions:

- Once a new visible event occurs, he infers new information about what has happened from his knowledge of the system.
- If this information conforms with his previous deductions, then it does not signal information flow.
- The fact that this new information differs from his previous deductions signals information flow.

Within this framework, the “trivial piece of code” int high; read(high); write(2) would not leak any information since at no point before or after observing the output 2 can Larry deduce any information about the value input in high, apart from his (Larry’s) initial knowledge that high is an int! As a corollary, if Harry has no
choice, then his activity should not be labeled as information flow. In other words, information flow is only aposteriori: it is meant to model situations in which Larry learns about Harry’s choices that precede Larry’s deductions.

Formally, in our setting, Larry’s knowledge at some instant \( T \) is modeled by an observed behavior \( w \) together with the set of possible traces in the system that may be observed as \( w \). The deduction mechanism is simply the inverse of the mapping which transforms a trace of the system into an observed behavior. We then say that the system has information flow if the knowledge at some point \( T \) is contradicted at some further point \( T' \), in the sense that \( \text{some of the traces which were thought as possible at } T \text{ are no longer possible as prefixes of traces at } T' \). This gives a simple and elegant definition of information flow, stating that prefix closure is “compatible” with Larry’s knowledge.

One of the two central results of our paper is that the two alternative views, the nondeterministic noninterference and the deducible information flow, are equivalent. This fundamental result says, in fact, that whatever Harry considers secure cannot be “broken” by Larry, and vice-versa.

Our system model is of a system accepting (generating) only finite traces. The actual restriction is in fact more subtle, and can be traced in the proof of the equivalence theorem: Harry is never left with the possibility to make infinitely many invisible actions. We do not have a proof of the equivalence theorem in the absence of this assumption. But note that no realistic model of information flow could accept Harry making infinitely many actions without letting Larry observe this. This restriction is related to the exclusion of the so-called Zeno behaviors in real-time systems (see e.g. [BP00]), that is, behaviors in which Harry makes infinitely many moves in finite amount of time. We therefore call strategies in which Harry can make infinitely many actions as Zeno strategies and exclude them from the model.

Of course, a more proper study of this situation necessitates a dense real-time semantics, which is out of the scope of this paper.

We show that our model is strictly weaker than e.g. the Perfect Security Property [ZL97], or the different types of noninterference [GM82, Man00]. We also show that absence of information flow in a finite-state system is decidable, and provide an exponential-space, linear time algorithm for this problem. The algorithm goes doubly exponential if nondeterministic finite-state automata are used as system models.

We also show that our approach also handles information flow through timing. The idea is rather standard: each high-level event that has a non-zero duration is mapped to a number of “clock ticks” representing the duration of the event. The current setting handles only discrete timing, but using ideas that are standard in models of real-time systems like timed automata [AD94] this can be generalized to a continuous time domain. It might however be the case that the decidability result will not hold within a timed automata setting.

A final difficulty in obtaining the equivalence theorem concerns the the system decisions allowing or not Harry to issue or not an action. In the \( 1/2 \)-like “game model” for nondeterministic noninterference this situation is handled properly by the use of strategies. It is not the case for the trace-based deduction model, which is not strategy-based. To correct this, we insert a new observable event \( \top \) just before each visible action, an event which models the possibility that the system has to forbid Harry from making any choice. This amounts to making visible the part in each strategy which represents system decisions. (Note that this “hack” is similar, in some sense, to the insertion of “clock ticks” for handling information flow through timing.) We also provide an alternative, slightly less elegant notion of deducible information flow which allows the handling of such situations without the “hack” of inserting visible decisions of the system, and prove that the the new notion is equivalent with the introduction of the visible \( \top \).

We think that the problem mentioned above is not a problem with our principle of defining information flow through deducibility, but rather with the ambiguity of nondeterminism inherent in trace models. We believe that a proper, twoplayer game-theoretic restatement would be able to capture this situation more clearly.

The paper goes on as follows: in the next section we give the two notions of information flow, together with some simple examples supporting the intuition. We also discuss the problem posed by handling system’s choices between permitting or disallowing Harry’s actions within the deducibility-based model. We also show that our model can capture information flow through timing. We then prove the main theorem of this paper in the third section. In the fourth section we generalize our definition to the case where some of the high-level actions are “not interesting” for Larry, and prove that this case reduces to the particular case of section 2. The fifth section serves for showing that our definition of information flow is strictly weaker than Perfect Security Property, Separability, Generalized Noninterference and Forward Correctability (see [ZL97, Man00, McL94]). In the sixth section we show that the problem of deciding whether a finite-state system has information flow is decidable. We end with a section with conclusions and directions of future research.

2 The trace model

Throughout the paper we will assume that the set of events which may occur through the behavior of the sys-
2.1 Nondeterministic noninterference

Our restatement of noninterference necessitates redefining the type of choices that Harry can do – namely, we consider choices as strategies for interacting with the system. The idea is that Harry cannot define beforehand what he’s going to do at each moment, since his choices may not be validated by the system. Hence, he must design a strategy for his interaction with the system depending on the system decisions about what is allowed and what is not.

The intuition is the following: for each $w \in E^*$, Harry must define his choice to act after the system performed as $w$. Harry must also take into consideration situations in which the system does not allow him to act in any way. The formalization is the following:

**Definition 1** A strategy for Harry for interacting with a system $T \subseteq E^*$ is a mapping $s: E^* \rightarrow \text{Inv} \cup \{ \bot \}$ satisfying the following properties:

- For each $w \in E^*$, if $s(w) = e \in \text{Inv}$ then $w \in T$.
- For each $w \in E^*$, if $s(w) = \bot$ then there exist some $e \in \text{Vis}$ with $w \in T$.

The two requirements are called the consistency requirements for the strategy $s$.

Hence, for $w \in E^*$, $s(w) \in E$ denotes the fact that the system lets Harry choose an invisible move, whereas $s(w) = \bot$ denotes the fact that the system forbids Harry to choose any move.

The set of behaviors spawned by a strategy is the set of behaviors that are produced according to Harry’s decisions encoded in the strategy:

$\text{Beh}(s) = \{ t \in \text{Tr} | \forall i \leq n, \quad \begin{align*}
&\text{if } s(t[1..i]) = \bot \text{ then } t[i+1] \in \text{Vis} \text{ or } i = n \\
&\text{if } s(t[1..i]) \in \text{Inv} \text{ then } n > i \text{ and } t[i+1] = s(t[1..i]) \}
\}$

Note that, in order to build $\text{Beh}(s)$ for some strategy $s$ we first have to build the set inside the pref operator. In this construction, only the system is allowed to decide when it stops – fact which is modeled by the last condition. Remind also that we have considered here only systems accepting finite behaviors – see also the discussion on Zeno strategies below.

Note also that, in the behavior spawn by a strategy, only traces in $\text{Tr}$ matter, so when defining a strategy we only need to specify it on subsets that construct the set of behaviors of the strategy.

**Example 1** Consider the system $T = \text{pref} \{ h_1 l_1, h_2 l_2 \}$, in which $\text{Inv} = \{ h_1, h_2 \}$ and $\text{Vis} = \{ l_1, l_2 \}$. The two “essentially different” strategies for Harry in $T$ are $s_1(\varepsilon) = h_1$, $s_1(h_1) = \bot$, resp. $s_2(\varepsilon) = h_2$, $s_2(h_2) = \bot$. The sets of behaviors are respectively $\text{Beh}(s_1) = \text{pref} \{ h_1 l_1 \}$ ($i = 1, 2$).

**Example 2** Consider now the system $T' = \text{pref} \{ h_1 l_1, h_1 l_2, h_2 l_1, h_2 l_2 \}$ with the same sets of visible and invisible events. Here again Harry has only two “essentially different” strategies: $s_1(\varepsilon) = h_1$, $s_1(h_1) = \bot$ and $s_2(\varepsilon) = h_2$, $s_2(h_2) = \bot$. The sets of behaviors are respectively $\text{Beh}(s_1) = \text{pref} \{ h_1 l_1, h_2 l_2 \}$ ($i = 1, 2$).

**Example 3** Consider now the system $T_{\text{zeno}} = \text{pref} \{ h^n | m, n \geq 0 \}$, with $E = \{ h \}$, $\text{Low} = \{ l \}$, $o(h) = \varepsilon$, $o(l) = l$. As a first example, $h^n \bot h^m \bot$ denotes the following strategy$^3$

$$s_1(h^i) = \begin{cases}
  h & \text{if } i < n \\
  \bot & \text{if } i = n \\
  h & \text{if } i < m \\
  \bot & \text{if } i = m
\end{cases}$$

The set of behaviors is

$$\text{Beh}(s_1) = \{ h^k, h^n \bot h^l | k \leq n, 0 \leq l \leq m \}$$

$^3$We denote here strategies using classical notations from regular expressions.
In strategy \( s_1 \), the system allows Harry to “make his choice” \( n \) times, then dissallows any choice, – and hence the next action has to be \( l \) – then allows again Harry to make \( m \) choices, then again dissallows any choice, fact which terminates system execution.

**Example 4** In the framework of Example 3, \( s_{\text{zeno}} = h^n \perp h^k \) denotes the following strategy for Harry:

\[
s_{\text{zeno}}(h^i) = \begin{cases} h & \text{if } i < n \\ \bot & \text{if } i = n \\ s_{\text{zeno}}(h^n \perp h^k) = h, \forall k \geq 0 \end{cases}
\]

Strategy \( s_1 \) says that the system allows Harry to “make his choice” \( n \) times, then dissallows any choice for the next move then Harry is again left with making his choice forever.

The problem with this strategy is that the system permits Harry to decide to run forever. This is a strategy that we will dissallow in the following, for two reasons:

1. Infinite behaviors cannot be properly observed.
2. In any realistic model, unboundedly many actions should eventually make time progress with an unboundedly amount of time – in order to avoid the so-called Zeno phenomenon.

In our framework, the set of behaviors for \( s_{\text{zeno}} \) cannot contain any trace of the form \( h^n \perp h^i \): the only object that is “generated” by the strategy and hence could produce such prefixes in \( \text{Beh}(s_{\text{zeno}}) \) is the infinite word \( w = h^n \perp h^\omega \).

But, by definition, \( \text{Tr} \subseteq E^* \), hence \( w \) is not a member of \( \text{Tr} \), and \( \text{Beh}(s_{\text{zeno}}) \) cannot contain prefixes of an infinite word.

We consider here systems accepting only finite behaviors, in order to avoid the so-called Zeno phenomenon [BP00] for invisible behaviors – that is, the fact that infinitely many invisible actions take place but Larry doesn’t observe that. A more proper treatment of this phenomenon would require considering a dense time semantics for systems, which is out of the scope of our paper. Note also that such a dense time semantics would clarify the handling of information flow through timing in our framework.

**Definition 2** A strategy \( s \) is **non-Zeno** if there exists no infinite sequence of words \( (t_n)_{n \geq 0} \) such that

\[
s(t_n) \in \text{Inv} \text{ and } t_{n+1} = t_n s(t_n)
\]

A strategy is **Zeno** for \( t_0 \) if it satisfies the negation of the above property, with the sequence of words starting in \( t_0 \).

**Remark 1** Note that for each strategy which is Zeno for some trace \( t_0 \), no trace that extends \( t_0 \) can be in \( \text{Beh}(s) \), that is, for all \( t \geq t_0 \), \( t \notin \text{Beh}(s) \).

To show this property, observe that, in order to compute \( \text{Beh}(s) \), we first have to compute the set

\[
A_s = \{ t \in \text{Tr} \mid \forall i \leq n, \begin{cases} \text{if } s(t[1..i]) = \bot \text{ then } t[i+1] \in \text{Vis} \text{ or } i = n \\ \text{if } s(t[1..i]) \in \text{Inv} \text{ then } n > i \text{ and } t[i+1] = s(t[1..i]) \}
\}
\]

In particular, for \( t_0 \) to belong to \( \text{Beh}(s) \), we must have some finite sequence \( \tau = t_0 a_1 \ldots a_n \) satisfying the property that, for all \( 0 \leq i \leq n \),

\[
\begin{align*}
&\text{if } s(t_0 a_1 \ldots a_i) \in \text{Inv} \text{ then } n > i \text{ and } a_{i+1} = s(a_1 \ldots a_i), \\
&\text{if } s(t_0 a_1 \ldots a_i) = \bot \text{ then } a_{i+1} \in \text{Vis} \text{ or } i = n
\end{align*}
\]

By the Zeno assumption for \( s \), we may prove by induction that \( s(t_0 a_1 \ldots a_i) \in \text{Inv} \). But then, the only object that satisfies the first property is the infinite sequence \( t_0 s(t_0) s(t_0 a_1) \ldots \), which is not a member of \( \text{Tr} \) by hypothesis (\( \text{Tr} \) models only finite behaviors). Therefore, \( t_0 \notin \text{Beh}(s) \) and, similarly \( t \notin \text{Beh}(s) \) for any \( t \geq t_0 \).

With strategies meant to model high-level choices, non-interference should be restated as follows:

**Distinct strategies cannot be distinguished by observation.**

For example, the system \( \text{Tr} \) in Example 1 above should not satisfy noninterference since the trace \( h_1 l_1 \in \text{Beh}(s_1) \) gives the observation \( l_1 \) and there is no trace in \( \text{Beh}(s_2) \) that gives the same observation. On the other hand, \( \text{Tr}' \) from Example 2 should satisfy noninterference since for both \( i = 1, 2 \), there exists a trace that gives the observation \( l_i \) in both \( \text{Beh}(s_1) \) and \( \text{Beh}(s_2) \). The formalization is the following:

**Definition 3** A system \( \text{Tr} \) has nondeterministic noninterference if for each two strategies \( s_1, s_2 \)

\[
\text{(NNI) } o(\text{Beh}(s_1)) = o(\text{Beh}(s_2))
\]

Or, alternatively, for each \( t \in \text{Beh}(s_1) \) there exists \( t' \in \text{Beh}(s_2) \) with \( o(t_1) = o(t_2) \).

### 2.2 Deducible information flow

So far, we have seen Harry’s definition of “information security”. The next step is to define Larry’s view on it. This is based on what can Larry deduce by observing system evolution.

To this end we will make use of the **low-level equivalent set** from [ZL97], denoted \( \text{LLES}(w, \text{Tr}) \), which stands for the set of traces in the system that produce the same observation as \( w \) (from Larry’s point of view):}

\[
\text{LLES}(w, \text{Tr}) = \{ w' \in \text{Tr} \mid o(w) = o(w') \}
\]
This is generally considered to belong to the knowledge of Larry after each occurrence of the trace $w$. As we already noted in the introduction, this is equivalent to the fundamental assumption about Larry’s apriori knowledge that he knows the whole specification of $Tr$. Since the set of traces of the system is prefix closed, we consider that Larry’s knowledge after the occurrence of $w$ is composed of all the prefixes of the traces that give the same observation. We then denote $\text{knl}(w, Tr)$ Larry’s knowledge after the behavior $w$ has passed in the system $Tr$:

$$\text{knl}(w, Tr) = \{ z \mid z \leq w \text{ for some } w' \in \text{LLES}(w, Tr) \} = \text{pref}(\text{LLES}(w, Tr))$$

Note that $\text{knl}(w, Tr) = \text{knl}(w', Tr)$ for any $w, w' \in Tr$ with $o(w) = o(w')$.

**Definition 4** The system represented by $Tr$ has no (deducible) information flow if for all $w, w' \in Tr$ such that $w \leq w'$, we have that:

$$\text{knl}(w, Tr) \subseteq \text{knl}(w', Tr).$$

$Tr$ has information flow if there exists $w, w' \in Tr$ with $w \leq w'$ for which the inclusion is not satisfied.

In other words, in a system having information flow, Larry can pass from a state in which he cannot rule out some traces, to a state in which he deduces that certain traces couldn’t have previously occurred.

**Example 5** Consider again the following system $Tr = \text{pref}\{h_1l_1, h_2l_2\}$ from Example 1. In the system $Tr$ we have information flow because:

$$\text{knl}(h_1, Tr) = \text{pref}\{h_1, h_2\} \not\subseteq \text{knl}(h_1l_1, Tr) = \text{pref}\{h_1l_1\}.$$ 

This corresponds to the following intuition. Suppose Larry had observed $\varepsilon$. In this moment he knows that the set of possible traces is $\{\varepsilon, h_1, h_2\}$. When he sees $l_1$, he deduces that $h_2$ could not have been possible at his previous observation.

**Example 6** Consider now the system $Tr' = \text{pref}\{h_1l_1, h_1l_2, h_2l_1, h_2l_2\}$, within the same framework as above. In the system $Tr'$ we do not have information flow because for all $i, j \leq 1, 2$:

$$\text{knl}(h_i, Tr') = \text{pref}\{h_1, h_2\} \not\subseteq \text{knl}(h_i l_i, Tr') = \text{pref}\{h_1l_j, h_2l_j\}.$$

**Example 7** Finally, consider $Tr'' = \text{pref}\{h_1l_1, h_2l_2\}$, with $L = \{h_1, h_2\}$ and $V = \{l_1, l_2\}$. The system $Tr''$ too has no information flow since for all $i = 1, 2$:

$$\text{knl}(h_i, Tr) = \text{pref}\{h_1, h_2\} \not\subseteq \text{knl}(h_i l_i, Tr) = \text{pref}\{h_1l_i, h_2l_i\}.$$ 

This example corresponds to the “trivial program” from Introduction.

**Example 8** We may prove that information flow through termination is also properly handled in our framework: consider $Tr_{term} = \text{pref}\{h_1l_1, h_2l_2\}$, within the same framework as in the previous example. Then:

$$\text{knl}(h_1, Tr) = \text{pref}\{h_1, h_2\} \not\subseteq \text{knl}(h_1l_1, Tr) = \text{pref}\{h_1l_1\}$$

This is in concordance with the intuition that the impossibility to continue $h_2l_2$ with $l$ must be information leak.

An alternative for Definition 4 is stated in the following:

**Proposition 1** The system represented by $Tr$ has no information flow if and only if the following property holds:

- (DIF) For all $t, t' \in Tr$ and $u \in E^*$ with $o(t) = o(t')$ and $o(u) \in Vis$, if $tu \in Tr$ then there exists $w' \in E^*$ such that $o(u') = o(u)$ and $t'w' \in Tr$.

**Proof:** The fact that Definition 4 implies property (DIF) is rather straightforward, since, under the assumption that $t, tu \in Tr$, the inclusion $\text{knl}(t, Tr) \subseteq \text{knl}(tu, Tr)$ implies that for any $t' \in Tr$ with $o(t) = o(t')$, $t' \in \text{knl}(tu, Tr)$, which in turn is equivalent to the existence of $u'$ such that $o(u) = o(u')$ and $t'u' \in Tr$.

For the reverse implication, note that it suffices to prove that for all $w, w' \in Tr$, with $w \leq w'$, $\text{LLES}(w, Tr) \subseteq \text{knl}(w', Tr)$. The case $o(w) = o(w')$ is trivial since it implies $\text{knl}(w, Tr) = \text{knl}(w', Tr)$.

Suppose that $o(w) < o(w')$, so $o(w') = o(w)z_1 \ldots z_n$, for some $n \geq 1$ and $z_1, \ldots, z_n \in Vis$. Then there exist $v_1, \ldots, v_n \in E^*$ such that $w' = wv_1 \ldots v_n$ and $o(v_i) = z_i$, for all $1 \leq i \leq n$. Take also $w'' \in \text{LLES}(w, Tr)$.

We may first apply the hypothesis (DIF) for $t = w$, $t' = w''$, $u = v_1$ to obtain that there exists $v_1' \in E^*$ such that $o(v_1') = o(v_1)$ and $w''v_1' \in Tr$. Now, for $t = wv_1$, $t' = w''v_1'$ and $u = v_2$, we will obtain that there exists $v_2' \in E^*$ such that $o(v_2') = o(v_2)$ and $w''v_1'v_2' \in Tr$. By induction on the remaining $v_i$s, we get $v_1, \ldots, v_n \in E^*$ such that $w''v_1' \ldots v_n' \in Tr$ and $o(v_i) = o(v_i')$ for all $1 \leq i \leq n$. The latter identities imply that $o(w''v_1' \ldots v_n') = o(w')$, and hence $w''v_1' \ldots v_n' \in \text{LLES}(w', Tr)$. As a consequence, $w'' \in \text{knl}(w', Tr)$.

**2.3 Making system decisions visible**

Consider the system $Tr_{limit} = \text{pref}\{h_1l_1, h_2l_2, l_2\}$, where $L = \{h\}$ and $V = \{l_1, l_2\}$. According to Definition 4, this system has no information flow:

$$\text{knl}(\varepsilon, Tr) \subseteq \text{knl}(l_1, Tr) = \text{pref}\{h_1l_1\}$$

and

$$\text{knl}(\varepsilon, Tr) \subseteq \text{knl}(l_2, Tr) = \text{pref}\{h_2l_2\}$$

On the other hand, according to Definition 3, system $Tr_{limit}$ does not satisfy nondeterministic noninterference:
consider the following two strategies: \( h \downarrow \), resp. \( \perp \), that is,
\[
s_1(\varepsilon) = h_1, \quad s_1(h_1) = \perp, \quad s_2(\varepsilon) = \perp
\]
We then have
\[
\text{Beh}(s_1) = \text{pref}\{hl_1, hl_2\} \quad \text{Beh}(s_2) = \text{pref}\{l_2\}
\]
and hence the trace \( t = hl_1 \) cannot be observed in the second strategy.

The point about this difference is the inability of our deduction model to separate what is Harry’s choice from what is system’s choice. This separation is made possible in the definition of nondeterministic noninterference by means of specifying jointly a strategy for Harry combined with a strategy for the system.

We show here that, by slightly modifying our setting, we are able to capture such information flow. The basic idea is simple and rather straightforward: we just need to make system choices visible. That is, whenever the system disallows Harry to make any choice, this fact should be signaled to Larry. The modification is neither more complicated nor simpler than the insertion of clock ticks, and it can be done in two ways, as we show in this subsection.

First, we could imagine the following type of observability: whenever a new observable action occurs, the observation is divided in two steps:

1. first, Larry is informed that a new observable action occurred (a system decision).
2. then he is informed of the actual observable value.

We will call observations of the first kind as limit observations, since they model what Larry could observe “just before” a new event occurs. In such a particular setting, our system \( \text{Tr}_{\text{limit}} \) has information flow:

- On the very moment when \( l_1 \) occurs, Larry is informed that a new observable event occurred, but, since he does not know which one it is, he deduces that \( \{\varepsilon, h\} \) may have occurred before.
- When he is informed that the new event is \( l_1 \), he concludes that his previous deduction was incorrect, since only \( \varepsilon \) could have occurred before \( l_1 \).

The alternative is to append a new observable event \( \top \) which will precede all observable events in \( \text{Tr} \) — that is, transform \( \text{Tr}_{\text{limit}} \) into \( \text{Tr}'_{\text{limit}} = \{h \top l_1, h \top l_2, l_1 l_2\} \). The new system \( \text{Tr}'_{\text{limit}} \) has information flow according to our setting from Section 2:

\[
\text{knl}(\top, \text{Tr}) = \{\varepsilon, h, h \top, \top\} \not\subseteq \text{knl}(\top l_1, \text{Tr}) = \{\varepsilon, h, h \top\}
\]

These remarks can be generalized as follows:

**Definition 5** A system \( \text{Tr} \) has no information flow in the limit if

1. The system has no deducible information flow and
2. For each \( t \in \text{Tr} \) and \( z \in \text{Vis} \) with \( tz \in \text{Tr} \), the following inclusion holds:

\[
\{t'z : t' \in \text{LLES}(t, \text{Tr}) \mid \exists z' \in \text{Vis} \text{ such that } t'z' \in \text{Tr} \}
\]

The insertion of \( \top \) events, which model limit observations, is formally defined as follows: consider the mapping \( T : E \rightarrow (E \cup \{\top\})^* \) defined by \( T(e) = \top \) if \( e \in \text{Vis} \) and \( T(e) = e \) otherwise. As usual, we may extend \( T \) to traces, and hence we get the transformed system \( T(\text{Tr}) = \{T(t) \mid t \in \text{Tr}\} \). We will consider \( \top \) an observable event, hence work with \( \text{Vis}' = \text{Vis} \cup \{\top\} \).

We may then prove the following:

**Proposition 2** \( \text{Tr} \) has no information flow in the limit if and only if the system \( T(\text{Tr}) \) has no deductible information flow.

*Proof:* Suppose first that \( T(\text{Tr}) \) has no deductible information flow. To prove that \( \text{Tr} \) has no deductible information flow, we just have to notice that

\[
\text{knl}(T(t), T(\text{Tr})) = T(\text{knl}(t, \text{Tr}))
\]

and for any two subsets \( K_1, K_2 \subseteq \text{Tr} \),

\[
K_1 \subseteq K_2 \Leftrightarrow T(K_1) \subseteq T(K_2)
\]

In order to prove property 2 from Definition 5, let \( t, t' \in \text{Tr} \), \( z \in \text{Vis} \) and \( z' \in \text{Vis} \) such that \( tz \in \text{Tr} \), \( o(t) = o(t') \) and \( t'z' \in \text{Tr} \).

We consider \( w = T(t) \) and \( w' = T(t') \) which are traces in \( T(\text{Tr}) \). Clearly, \( o(w) = o(w') \).

Since \( tz \in \text{Tr} \) and \( t'z' \in \text{Tr} \), we obtain that \( T(tz) = w \top z \in T(\text{Tr}) \) and \( T(t'z') = w' \top z' \in T(\text{Tr}) \). We apply (DIF) for \( w' \top ^* \in T(\text{Tr}) \) and \( z \in \text{Vis} \cup \{\top\}^* \) and we obtain that there exists \( u' \in (E \cup \{\top\})^* \) with \( o(u') = z \) such that \( w' \top u' \in T(\text{Tr}) \). Also, by the definition of \( T \), we have that \( u' = za \) for some \( a \in \text{Inv}^* \). Hence, \( w' \top z \in T(\text{Tr}) \) which implies \( t'z' \in \text{Tr} \).

For the reverse implication, suppose that \( \text{Tr} \) has no information flow in the limit. Let \( w, w' \in T(\text{Tr}) \) and \( s \in (E \cup \{\top\})^* \) such that \( o(w) = o(w') \), \( o(s) \in \text{Vis} \cup \{\top\} \) and \( ws \in T(\text{Tr}) \). If we prove that there exists \( s' \in (E \cup \{\top\})^* \) such that \( o(s) = o(s') \) and \( ws' \in T(\text{Tr}) \) then, by Proposition 1 we get that \( T(\text{Tr}) \) has no deductible information flow.

By deleting \( \top \) events from \( w \) and \( w' \), we get \( t, t' \in \text{Tr} \) with \( T(t) = w, T(t') = w' \) and \( o(t) = o(t') \). Similarly, we get \( u \in E^* \) for which \( T(u) = s \) and \( tu \in \text{Tr} \).

We have two cases:
1. if \( o(s) = T \), we must have \( s = \tau T \) with \( \tau \in \text{Inv}^+ \). So, by the fact that \( w \tau T \in T(Tr) \), we have that there exists \( z \in \text{Vis} \) such that \( w \tau T z \in T(Tr) \). Hence, \( t \tau z \in Tr \).

We apply (DF) for \( t \tau \), \( t' \), \( z \) and we obtain that there exists \( u' \in E^+ \) such that \( o(u') = z \) and \( t' u' \in Tr \). Since \( u' = \alpha z \beta \), for some \( \alpha, \beta \in \text{Inv}^+ \), we get that \( t' \alpha z \in Tr \). Consequently, \( u' \alpha z \in T(t' \alpha z) \in T(Tr) \). So, for \( s' = \alpha T \) we have \( w' s' \in T(Tr) \) and \( o(s') = o(s) \), which ends the proof in this first case.

2. if \( o(s) = z \in \text{Vis} \), we have that \( s = z \tau \), for some \( \tau \in \text{Inv}^+ \), \( w = \sigma \tau \), for some \( \sigma \in (E \cup \{T\})^+ \) and \( tz \in Tr \). Moreover, since \( o(w) = o(w') \), we obtain that \( w' = \sigma' \tau \), for some \( \sigma' \in (E \cup \{T\})^+ \). By the definition of \( T \), this implies that there exists \( z' \in \text{Vis} \) such that \( w' z' \in T(Tr) \) and consequently, \( t' z' \in Tr \).

We apply the second property from Definition 5 for \( t, t', z \), and we obtain that \( t' z \in Tr \). Hence, there exists \( s' = z \) such that \( w' s' \in T(Tr) \) and \( o(s) = o(s') \).

\[ \square \]

### 2.4 Information flow through timing

In this final subsection we show how to use our framework to model information leak through timing.

The main idea is that the observation mapping \( o \) may also be used to model information flow through timing by putting \( o(a) = \sqrt{a} \), with \( \sqrt{a} \in \text{Vis} \) modeling time passage, for each action \( a \in \text{Inv} \) which is supposed to be time consuming. Of course, this setting supports only discrete timing, dense timing would require a timed words model [ACM02], which will not be discussed in this paper.

**Example 9** Consider the system \( T_{Tr} = \text{pref}\{h_1 l, h_2 h_1 l\} \). If both \( h_1 \) and \( h_2 \) have equal duration, \( T_{Tr} \) leaks information since by observing the interval between system start and the occurrence of \( l \), Larry may deduce whether Harry’s first choice was \( h_1 \) or \( h_2 \).

With our insertion of \( \sqrt{a} \) the system changes to \( T_{Tr} = \text{pref}\{h_1 \sqrt{l}, h_2 \sqrt{h_1 \sqrt{l}}\} \), which has information flow, since \( \text{knl}(h_1, Tr) = \text{pref}\{h_1, h_2\} \subseteq \text{knl}(h_2 \sqrt{h_1 \sqrt{l}}, Tr) = \text{pref}\{h_2 \sqrt{h_1 \sqrt{l}}\} \).

The general discrete case, that is, of unobservable actions having different durations, could be easily handled by “decomposing” each action into time-unit actions and applying the above ideas. Hence, if the duration of \( h_1 \) is \( d(h_1) = 2 \) and \( d(h_2) = 1 \), then we transform the system \( T_{Tr} \) into \( T_{Tr} = \text{pref}\{h_1 h_1 l, h_2 h_1 h_1 l\} \) which leaks information about Harry’s initial choice between \( h_1 \) and \( h_2 \).

### 3 Relating nondeterministic noninterference and deducible information flow

We will show here that the two notions are equivalent, which is one of the two central results in this paper. We will actually show the following:

**Theorem 1** \( Tr \) satisfies nondeterministic noninterference if and only if \( T(Tr) \) has no deducible information flow.

We will first prove the following:

**Lemma 1** \( Tr \) satisfies nondeterministic noninterference if and only if \( T(Tr) \) satisfies nondeterministic noninterference as well.

**Proof:** By straightforward modification of strategies of \( Tr \) into strategies of \( T(Tr) \) — each strategy \( s \) in \( Tr \) can be transformed into a strategy \( s^\top \) of \( T(Tr) \) defined as follows:

\[
\forall w \in E^+, \begin{cases} s^\top(T(w)) = s(w) \\ s^\top(T(w) \top) = \bot \end{cases}
\]

By trivial verification we get that the mapping \( T \) is correctly defined on strategies, that is,

1. For each strategy \( s \) on \( Tr, T(s) \) is a strategy on \( T(Tr) \)
2. And for each strategy \( s \) on \( T(Tr) \), there exists a strategy \( s' \) on \( Tr \) such that \( s = T(s') \).

It is then easy to show that \( \text{Beh}(s^\top) = T(\text{Beh}(s)) \). But \( o \) and \( T \) commute, that is, for each \( w \in E^+ \), \( o(T(w)) = T(o(w)) \). Then all we have to see is that

\[
o(\text{Beh}(s^\top)) = o(T(\text{Beh}(s))) = T(o(\text{Beh}(s)))
\]

This implies that

\[
o(\text{Beh}(s_1)) = o(\text{Beh}(s_2)) \iff o(\text{Beh}(s_1^\top)) = o(\text{Beh}(s_2^\top))
\]

fact which ends our proof. \( \square \)

**Proof:** [of Theorem 1] Thanks to the previous lemma, we will actually prove that \( T(Tr) \) satisfies nondeterministic noninterference if and only if \( T(Tr) \) has no deducible information flow.

Suppose \( T(Tr) \) has deducible information flow. According to Proposition 1, there exist \( t_1, t_2 \in T(Tr) \) and \( u \in E^+ \) with \( o(t_1) = o(t_2) \), \( o(u) \in \text{Vis} \) and \( t_1 u \in Tr \) such that for all \( u' \in E^+ \) with \( o(u') = o(u) \) we have \( t_2 u' \notin Tr \).

We will construct then two strategies \( s_1 \) and \( s_2 \) from \( t_1 u \), resp. \( t_2 \), which will infer nondeterministic noninterference. To this end, we enumerate the events in \( t_1, t_2 \) and \( u \),...
say \( t_i = t_{i1} \ldots t_{in} \) for some \( n_1, n_2 \geq 0 \) and \( u = u_1 \ldots u_n \). Then \( s_1 \) and \( s_2 \) are defined as follows:

\[
\forall j = 1, 2, \forall 1 \leq i < n_j, s_j(t_{ij} \ldots t_{in}) = \begin{cases} 
  t_{ij} & \text{if } t_{ij+1} \in \text{Inv} \\
  \bot & \text{otherwise}
\end{cases},
\]

\[
\forall 0 \leq i \leq n, s_1(t_{i1} \ldots t_{in1}) u_1 \ldots u_i = \begin{cases} 
  u_{i+1} & \text{if } u_{i+1} \in \text{Inv} \\
  \bot & \text{otherwise}
\end{cases},
\]

The above definitions give the form for \( s_1 \) and \( s_2 \) only on their “interesting” part – on arguments that are not of the specified form, \( s_1 \) and \( s_2 \) may give any result satisfying the requirements from Definition 3.

We claim that if \( o(ta) \not\in o(\text{Beh}(s_2)) \). To see this, note first that the only trace \( t'_2 \) in \( \text{Beh}(s_2) \) that has the same observation as \( t_1 \) is \( t_2 \), by uniqueness in the definition of \( s_2 \). Therefore, if we had some \( t'_2 \in \text{Beh}(s_2) \) with \( o(t'_2) = o(ta) \), the \( t'_2 \) should have \( t_2 \) as a prefix.

But, by hypothesis on \( t_1, u, t_2 \), we cannot have some \( t_2u' \in \text{Tr} \) with \( o(t_2u') = o(ta) \), which means that no \( t'_2 \) with \( o(t'_2) = o(ta) \) may belong to \( \text{Beh}(s_2) \), which proves the direct implication.

For the reverse implication, consider two strategies \( s_1 \) and \( s_2 \) satisfying the negation of the property (NNI), i.e. there exists \( t_1 \in \text{Beh}(s_1) \) such that for any trace \( t_2 \in \text{Beh}(s_2) \) we have \( o(t_1) \not\in o(t_2) \). Without loss of generality, we will take as \( t_1 \) the least trace, w.r.t. prefix ordering, that satisfies this property. Hence, if \( t_1 = ta \) for some \( a \in E \), \( t \) can be observed in \( s_2 \), that is, there exists some \( t' \in \text{Beh}(s_2) \) such that \( o(t') = o(t) \).

The theorem would be proved if we show that

Claim: \( t, a, t' \) is a triple that does not satisfy (DIF)

Remark 2 Note that a must be visible, or otherwise \( o(ta) = o(t') \) which would contradict the fact that \( t_3 = ta \) has no observable equivalent in \( \text{Beh}(s_2) \).

Remark 3 We may also prove, by induction, that the set \( \{ t' \in \text{Beh}(s_2) \mid o(t') = o(t) \} \) is totally ordered, because whenever we have \( t, t' \in \text{Beh}(s_2) \) with \( t' = t \) and \( o(\tau') = o(\tau) \), we get that \( h \in \text{Inv} \) and hence \( s_2(\tau) = h \), that is, \( \tau \) can only be “extended” in a unique way in \( \text{Beh}(s_2) \) with unobservable events.

Hence, we may choose \( t' \in \text{Beh}(s_2) \) to be the first trace, w.r.t. prefix ordering, which satisfies \( o(t') = o(t) \).

Suppose first that \( a \not\in \top \). Then, \( t = w \top \) by construction of \( \text{T}(\text{Tr}) \). Consequenty, \( t' = w \top \).

Furthermore, by construction of \( \text{T}(\text{Tr}) \), for any \( t'u' \in \text{T}(\text{Tr}) \), \( u' \) must start with a visible event, that is, whenever \( t'u' \in \text{T}(\text{Tr}) \) and \( o(u') = b \in \text{Vis} \) we must have also that \( t'b \in \text{T}(\text{Tr}) \).

In particular, if \( \text{T}(\text{Tr}) \) were to satisfy (DIF), then \( t'a \in \text{T}(\text{Tr}) \). But, by rewriting the defining property (NNI) for \( t_1 \), we get that \( t'a \not\in \text{Beh}(s_2) \). We then need to show that this also implies the stronger property that \( t'a \not\in \text{T}(\text{Tr}) \).

To this end, observe that, since \( t' \) ends with \( \top \), \( t' \) is always followed by a visible event, that is, there is no \( h \in \text{Inv} \) such that \( t'h \in \text{T}(\text{Tr}) \). Therefore we must have \( s_2(t') = \bot \). This means that after \( t' \), the system forbids Harry for choosing, and makes his own choice, which must be a visible event. But the construction of \( \text{Beh}(s_2) \) will always include such a continuation, that is, if \( t' \in \text{Beh}(s_2) \), \( s_2(t') = \bot \) and \( t'a \in \text{T}(\text{Tr}) \) then we should also have \( t'a \in \text{Beh}(s_2) \).

This means that we cannot have \( t'a \in \text{T}(\text{Tr}) \), which means that in fact the triple \( t, a, t' \) satisfies the negation of property (DIF) from Proposition 1, that is, \( \text{T}(\text{Tr}) \) has deducible information flow.

The proof will end if we show that \( a \) cannot be \( \top \), a result which relies on the Remark 1:

Suppose, for the sake of contradiction, that \( t_1 = t \top \), \( t' \in \text{Beh}(s_2) \) with \( o(t') = o(t) \) and for any \( u' \in E^* \) with \( o(u) = \top \) we have \( t'u' \not\in \text{Beh}(s_2) \). Remind also that we may choose \( t' \) to be the first, w.r.t. prefix ordering, that satisfies this property.

We will first prove that \( s_2(t') \not\in \top \). For, assuming the contrary, we would obtain that \( t'b \in \text{T}(\text{Tr}) \) for some \( b \in \text{Vis} \), and, since \( t' \top \not\in \text{T}(\text{Tr}) \) by assumption, we obtain that \( b \not\in \top \) either. But this would imply that \( t' = t \top \), which, on its turn, would imply that \( t_1 = t \top \) for some \( t \) with \( o(\tau) = o(\tau') \), which is forbidden by construction in \( \text{T}(\text{Tr}) \). Hence, \( s_2(t') \in \text{Inv} \).

We construct then an infinite sequence of events \( e = (e_n)_{n \geq 0} \) which represents the decisions coded in \( s_2 \) that extend \( t' \). That is, \( e_1 = s_2(t') \) and \( e_{n+1} = s_2(t'e_1 \ldots e_n) \). And we may easily observe, by induction, that \( e_i \in \text{Inv} \), for, otherwise, considering \( e_i \) the first event equal to \( \top \), and \( e_j \in \text{Inv} \) for all \( j < i \), we would conclude that \( o(t'e_1 \ldots e_{i-1}e_i) = o(t') \), which contradicts the assumption that \( t' \top \) has no equivalent observable in \( \text{Beh}(s_2) \).

But this means that \( s_2 \) is Zeno for \( t' \), which, by Remark 1, implies that \( t' \not\in \text{Beh}(s_2) \), in contradiction with our choice. This ends our proof of the Theorem 1. \( \square \)

4 Generalized deducible information flow

In this section we generalize our notion of deducible information flow to the case in which Larry is interested in information regarding only a part of Harry’s choices. We show here that information flow analysis in this generalized setting can be reduced to the formalism from Section 2.2.

The framework of our generalization is the following: we consider, as before, that the system under analysis is represented by a set of traces \( \text{Tr} \subseteq E^* \), where \( E = \text{Vis} \cup \text{Inv} \) is the set of possible events. Larry, the low-level user, will only be interested in making deductions about the high-level events from a subset \( \text{Conf} \subseteq \text{Inv} \). We will define a
deduction mapping $d : E \to \text{Conf} \cup \text{Vis} \cup \{\varepsilon\}$ by $d(e) = e$ if $e \in \text{Conf} \cup \text{Vis}$ and $d(e) = \varepsilon$, otherwise. We extend $d$ to sequences of events, as usual, by concatenating the result of $d$ applied to each event in the sequence. The fact that $d(e) = \varepsilon$ for some $e \in E$ signals that the occurrence the event $e$ is of no importance to Larry.

As an example, suppose that the “atomic” high level event $e$ amounts to the fact that Harry updates a certain file via some communication protocol. Larry may be interested only in getting information about the update operation, not in the type of communication protocol Harry uses. In this case, the mapping $d$ would simply “forget” the communication protocol.

Remind that $\text{LLES}(w, \text{Tr})$ was supposed to model the deductions Larry can make once he observes $o(w)$. In the current setting Larry is no longer interested in deducing all informations about Inv, but rather in getting informations about activities in Conf. Once the system issues trace $w$, Larry makes the following deductions:

$$\text{DED}(w, \text{Tr}) = \{d(w') \in \text{Tr} \mid o(w) = o(w')\}.$$ 

In other words, once the system behaves like $w$, Larry deduces all the projections onto $\text{Vis} \cup \text{Conf}$ of traces that give the same observation as $w$.

Continuing our parallel to Section 2.2, we define Larry’s generalized knowledge after seeing $w$, denoted by $\text{genkn}(w, \text{Tr})$, as the set of all prefixes of the deductions he can make after the system issues the trace $w$:

$$\text{genkn}(w, \text{Tr}) = \{z \mid z \preceq w' \text{ for some } w' \in \text{DED}(w, \text{Tr})\} = \text{pref}(\text{DED}(w, \text{Tr}))$$

The definition of deducible information flow in this generalized setting is obtained by replacing in Definition 4, $\text{kn}(w, \text{Tr})$ with $\text{genkn}(w, \text{Tr})$:

**Definition 6** The system represented by $\text{Tr}$ has no generalized deducible information flow w.r.t. the deduction mapping $d$ if for all $w, w' \in \text{Tr}$ such that $w \preceq w'$, we have that:

$$\text{genkn}(w, \text{Tr}) \subseteq \text{genkn}(w', \text{Tr}).$$

If there exist $w, w' \in \text{Tr}$ with $w \preceq w'$ such that the inclusion is not satisfied, we say that $\text{Tr}$ has generalized deducible information flow w.r.t. $d$.

Deducible Information flow is a particular case of generalized deducible information flow if we consider $\text{Conf} = \text{Inv}$. The essence of Theorem 2 below is that a certain type of reverse relationship also holds, namely that generalized deducible information flow reduces to “simple” deducible information flow.

**Theorem 2** $\text{Tr}$ has no generalized deducible information flow w.r.t. $d$ if and only if $d(\text{Tr}) = \{d(t) \mid t \in \text{Tr}\}$ has no deducible information flow.

**Proof:** In the system $d(\text{Tr})$, Larry’s knowledge for some trace $\alpha \in d(\text{Tr})$ where $\alpha = d(t)$, $t \in \text{Tr}$, is

$$\text{kn}(\alpha, d(\text{Tr})) = \{\beta \in d(\text{Tr}) \mid o(\alpha) = o(\beta)\} = \{d(t') \mid t' \in \text{Tr} \text{ and } o(d(t)) = o(d(t'))\} = \text{genkn}(t, \text{Tr})$$

Hence, the inclusion $\text{kn}(\alpha, \text{Tr}) \subseteq \text{kn}(\beta, \text{Tr})$, for some $\alpha, \beta \in d(\text{Tr})$ with $\alpha \preceq \beta$ is equivalent to the inclusion $\text{genkn}(t, \text{Tr}) \subseteq \text{genkn}(t', \text{Tr})$ for some $t, t' \in \text{Tr}$ with $t \preceq t'$ and $d(t) = \alpha, d(t') = \beta$.

The following is the equivalent of Proposition 1 for generalized deducible information flow:

**Proposition 3** $\text{Tr}$ has no generalized deducible information flow if and only if the following property holds:

(GDIF) For all $t, t' \in \text{Tr}$ and $u \in E^*$ with $o(t) = o(t'), o(u) \in \text{Vis}$, if $tu \in \text{Tr}$ there exists $t'' \in \text{Tr}$ and $u' \in E^*$ such that $o(u') = o(u), d(t') = d(t'')$ and $t''u' \in \text{Tr}$.

The proof of this property is similar to the proof of Proposition 1.

5 Relationship with other definitions of information flow

In this section we show how generalized deducible information flow can be related with other well-known information flow properties from the literature.

To this end we suppose, as in Section 4, that $E = \text{Vis} \cup \text{Inv}$, $\text{Conf} \subseteq \text{Inv}$ and $d$ is the deduction mapping.

5.1 The relation with Perfect Security Property and Separability

The Perfect Security Property (PSP, for short) from [ZL97] is defined for $\text{Conf} = \text{Inv}$. As we noticed earlier, in this context, generalized deducible information flow is exactly deducible information flow. We will prove that PSP implies deducible information flow.

We define $c : E \to \text{Conf} \cup \{\varepsilon\}$ by $c(e) = e$ if $e \in \text{Conf}$ and $c(e) = \varepsilon$, otherwise. We extend $c$ to handle sequences of events in the usual way.

**Definition 7** ([ZL97]) We say that a system $\text{Tr}$ satisfies the Perfect Security Property if the following properties hold:

1. $(\forall t \in \text{Tr}) : o(t) \in \text{Tr},$

2. $(\forall t_1, t_2 \in \text{Tr})(\forall h \in \text{Conf})(t_1 t_2 \in \text{Tr} \land c(t_2) = \varepsilon \land t_1 h \in \text{Tr} \Rightarrow t_1 t_2 h \in \text{Tr}).$
Example 10 Consider again the system from Example 7 in Section 2, $T'' = \text{pref}\{h_1, h_2\}$, with $\text{Vis} = \{l\}$ and $\text{Conf} = \text{Inv} = \{h_1, h_2\}$. Remind that the system $T''$ has no deducible information flow. But clearly $T''$ does not satisfy PSP since $o(h_1) \not\in T''$.

The next result will prove that our notion of deductible information flow is weaker than PSP. Moreover, following the example above, we obtain that it is strictly weaker than PSP.

Proposition 4 If a system $T$ satisfies PSP then it has no deductible information flow.

We rely on the following lemma which gives an equivalent formulation of PSP:

Lemma 2 A system $T$ satisfies PSP iff for all $t, u \in \Sigma^*$, $e, e' \in \Sigma^*$ such that $c(u) = e$ and $o(e) = o(e') = e$ if $teu \in T$ and $e' \in T$ then, $e'\!\ u \in T$.

Proof: [of Proposition 4] Suppose $T$ satisfies PSP. Let $t_1, t_2 \in T$ and $u \in E^*$ such that $o(t_1) = o(t_2)$, $o(u) = z \in \text{Vis}$ and $t_1u \in T$. We have to prove that there exists $u' \in E^*$ such that $o(u') = o(u)$ and $t_2\!\ u' \in T$.

Now, since $T$ satisfies PSP and $t_1u \in T$, we have that $o(t_1u) = o(t_1)o(u) \in T$. If $o(t_1) = l_1 \ldots l_n$ for some $n \in \mathbb{N}$ and $l_1, \ldots, l_n \in \text{Vis}$ then, $t_2 = h_1l_1 \ldots h_nl_nh_{n+1}$ for some $h_1, \ldots, h_{n+1} \in \text{Inv}^*$.

Now, if we apply the equivalent formulation of PSP from Lemma 2 for $t = e$, $e = e$, $e' = h_1$ and $u = l_1 \ldots l_nz$, we obtain that $h_1l_1 \ldots l_nz \in T$. Next, we apply the same property for $t = h_1l_1$, $e = e$, $e' = h_2$ and $u = l_2 \ldots l_nz$. We will obtain that $h_1h_2l_2 \ldots l_nz \in T$. Repeating the same process we will obtain that $t_2z \in T$. Hence we have found $u' = o(u)$ with the required properties.

Remark 4 Since separability ([McL94]) implies PSP, we also obtain that a system satisfying separability has no deducible information flow.

5.2 The relation with Generalized Noninterference and Forward Correctability

In this subsection we prove that our notion of generalized deductible information flow is weaker than Generalized Noninterference (GNI, for short). This property was first introduced in [McC87] and afterwards McLean offered in [McL94] a weaker version based on his interleaving functions framework.

GNI was originally defined in terms of high-level and low-level inputs and outputs but we can also define it in terms of confidential events, invisible and non-confidential events, and visible events, as in the MAKS framework [Man03].

Definition 8 We say that a system defined by a set of traces $T$ satisfies Generalized Noninterference (GNI), if for all $t_v \in T_v$, $t_c \in \text{Conf}^*$ and $\tau \in E^*$:

$$\tau \in \text{interleaving}(t_c, o(t_v)) \Rightarrow \exists t \in T_r \text{ such that } d(t) = \tau.$$

Example 11 We use again the system from Example 7 in Section 2, as an example of a system that satisfies our notion of generalized deductible information flow but does not satisfy generalized noninterference. The system under discussion is $T'' = \text{pref}\{h_1, h_2\}$, with $\text{Vis} = \{l\}$ and $\text{Conf} = \text{Inv} = \{h_1, h_2\}$. Remind that the system $T''$ has no generalized deductible information flow w.r.t. $d$ (since it has no deducible information flow). But $T''$ does not satisfy Generalized Noninterference since we may put $t_v = h_1l_1$, $t_c = \varepsilon$ and for the only element of interleaving($t_c, o(t_v)$) = $\{1\}$ $d^{-1}(1)$ = $\{1\}$ but $l \not\in T$.

We see again here an effect of the manifest assumption in Generalized Noninterference that at each moment Harry has the choice to do or not to do an action, which is relaxed by our notion of information flow.

Proposition 5 If the system $T$ satisfies GNI then, it has no generalized deductible information flow w.r.t. $d$.

Proof: Suppose $T$ satisfies GNI. Let $t_1, t_2 \in T$ and $u \in \Sigma^*$ such that $o(t_1) = o(t_2)$, $o(u) \in \text{Vis}$ and $t_1u \in T$. We have to prove that there exists $t'_2 \in T$ and $u' \in \Sigma^*$ such that $o(u') = o(u)$, $d(t_2) = d(t'_2)$ and $t'_2u' \in T$.

This can be obtained if we notice that $d(t_2) = d(t_2u) \in \text{interleaving}(c(t_2), o(t_1u))$.

Remark 5 Since forward correctability ([JT88]) implies GNI (see [Man03] for more details), we obtain that it also implies generalized deductible information flow w.r.t. $d$ as above.

6 A finite state model and its decidability

Throughout this section we assume that the system is represented by a finite automaton $A = (Q, E, \delta, q_0)$. Hence $T = L(A)$ is the set of traces which do not block the automaton (there is no final state since the language is prefix-closed). We also assume that all the states of the automaton are reachable, that is, for each $q \in Q$ there exists a trace $t \in \Sigma^*$ (in fact, $t \in T$!) that labels a path from $q_0$ to $q$.

In this setting, we are interested in solving the following decision problem:

Problem: [Finite-state information flow problem] Given an automaton $A$, does $L(A)$ have information flow?

Theorem 3 The finite state information flow problem is decidable.
The proof idea is the following: starting from the initial state \( q_0 \), we iteratively construct the set of states that are reachable by behaviors in \( \mathcal{T}r \) which give the same observation. This is done in a greedy manner, first constructing the set of states reachable by elements in \( \text{LLES}(a, \mathcal{T}r) \) for each \( a \in \text{Vis}, \) then by elements in \( \text{LLES}(ab, \mathcal{T}r) \) for each \( a, b \in \text{Vis}, \) etc. Note that at each step we obtain a finite set, and that each transition constructed this way is labeled by an element of \( \text{Vis} \) (which is finite by assumption). This process eventually ends as there are only finitely many subsets of \( Q \). We then only have to check, in the vein of Proposition 1, that for each reachable set of states \( T \subseteq Q \) and for each action \( a \in \text{Vis} \):

- either all states in \( T \) can issue a sequence of transitions whose observation is \( a \),
- or \( a \) cannot be the next event that can be observed after passing through some of the states in \( T \).

**Proof:** [of Theorem 3] We will consider here deterministic automata (in the sense of classical automata theory, see e.g. [HU92, Yu97]), that is \( \delta : Q \times E \rightarrow Q \cup \{ \} \). The uparrow denotes that \( \delta(q, a) \) is undefined — i.e. if the automaton receives the event \( a \) while in state \( q \), it is blocked forever.

We extend the transition function to words, that is, we build \( \overline{\delta} : (Q \cup \{\} \} \times E^* \rightarrow Q \cup \{\} \) by structural induction: for all \( q \in Q \), \( w \in E^* \), \( \overline{\delta}(q, w) = \overline{\delta}(q, w') = \overline{\delta}(q, w) \). We then only have to check, in the vein of Proposition 1, that for each reachable set of states \( \mathcal{S}_w \), we can construct the set of states that are reachable from \( S_{w_1} \) by \( \overline{\delta} \).

We define also the observable transition relation \( \theta \subseteq Q \times \text{Vis} \times Q \) as follows: for all \( q \in Q \), \( a \in \text{Vis} \),

\[
\theta(q, a) = \{ t \mid \exists t \in E^*, o(t) = a \text{ and } \overline{\delta}(q, t) = r \}
\]

We will then extend \( \theta \) to \( \overline{\theta} \) as follows:

\[
\overline{\theta}(S_w, a) = \bigcup_{q \in S_w} \theta(q, a)
\]

Since it might be the case that \( \overline{\delta}(q_0, a) = \overline{\delta}(q_0, a') \), we adopt the convention that \( \overline{\delta}(q_0, a) = \overline{\delta}(q_0, a') \).

It is easy to show that

\[
\forall w \in \text{Vis}^*, a \in \text{Vis}, \overline{\delta}(S_w, a) = S_{wa}
\]

Hence, we have an algorithm for the construction of a finite automaton \( B = (S, \overline{\theta}, Q_0, \overline{\theta}) \) where

\[
Q_0 = \{ q \in Q \mid \overline{\delta}(q_0, a) = q \text{ for some } a \in E^* \text{ with } o(a) = \epsilon \}
\]

**Claim:** \( L(A) \) has information flow iff there exists \( w \in o(\mathcal{T}r) \) with \( S_w \neq \{ \} \) and \( \overline{\delta}(q_0, w) = \overline{\delta}(q_0, w') \).

Note that the proof of this claim completes the proof of the decidability of the finite-state information flow problem since the family \( (S_w)_{w \in \mathcal{T}r} \) and the observable transition relation \( \theta \) can be constructed algorithmically.

To prove this claim, we rely on Proposition 1: since \( \mathcal{T}r \) is prefix closed, the property (DIF) is equivalent to the conjunction of the following two properties:

1. \( \delta(q_0, t'w') = \overline{\delta}(q_0, w) \), which means that \( q \in S_{tw} \), and
2. \( \delta(q_0, tw) \neq \overline{\delta}(q_0, w) \) by means of \( tw \in \mathcal{T}r \), hence \( \delta(q_0, tw) \notin S_{tw} \).

This ends our proof of the decidability of the finite-state information flow problem.

Note that the size of the automaton \( B \) is exponential in the size of the original automaton. This “state-space explosion” is even doubly-exponential if nondeterministic automata are used as system models.

### 7 Conclusions and future work

We have presented a framework for trace-based possibilistic information flow which starts from the principle that high-level choices should not be deduced by low-level users. Our framework labels less systems as having information flow, than other properties like Perfect Security Property, Separability, Generalized Noninterference or Forward Correctability, as it permits even more types of information flow than low-level users. We have shown that our model is sufficiently powerful to express only “partial information flow”, that is, the situation in which the low-level user is not interested in achieving information about all the high-level activity, but only in a part of it. And we have also designed, in the finite-state case, an exponential-space algorithm for deciding whether a system has information flow.

Several other directions can be taken to pursue this work. Firstly, we are developing a probabilistic version of our framework, together with an algorithm in the case of finite-state discrete Markov chains. Further development towards Markov decision processes is necessary too, similar to the game-theoretical framework whose need was apparent when discussing the ambiguity between internal and external choices. Secondly, we would like to adapt our approach to infinite-state systems. It is clear that, for such an approach to succeed, we may have to limit Larry’s deductive power, as it is the case with security protocols.

### References


