Theoretical Computer Science 631 (2016) 16–42

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

The complexity of synchronous notions of information flow security

Franck Cassez [a,b,1], Ron van der Meyden [b], Chenyi Zhang [b,*]

[a] Department of Computing, Macquarie University, Sydney, Australia
[b] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

ARTICLE INFO

Article history:
Received 24 March 2014
Received in revised form 18 September 2015
Accepted 8 March 2016
Available online 9 March 2016
Communicated by V. Sassone

Keywords:
Security
Information flow
Computational complexity

ABSTRACT

The paper considers the complexity of verifying that a finite state system satisfies a number of definitions of information flow security. The systems model considered is one in which agents operate synchronously with awareness of the global clock. This enables timing based attacks to be captured, whereas previous work on this topic has dealt primarily with asynchronous systems. Versions of the notions of nondeducibility on inputs, nondeducibility on strategies, and an unwinding based notion are formulated for this model. All three notions are shown to be decidable, and their computational complexity is characterised.

1. Introduction

Information flow security is concerned with the ability of agents in a system to deduce information about the activity and secrets of other agents. An information flow security policy prohibits some agents from knowing information about other agents. In an insecure system, an agent may nevertheless be able to make inferences from its observations, that enables it to deduce facts that it is not permitted to know. In particular, a class of system design flaws, referred to as *covert channels*, provide unintended ways for information to flow between agents, rendering a system insecure.

Defining what it is for a system to satisfy an information flow security policy has proved to be a subtle matter. A substantial literature has developed that provides a range of formal systems models and a range of definitions of security. In particular, in non-deterministic systems it has been found necessary to clarify the attack model, and distinguish between a passive attacker, which merely aims to deduce secret information from observations it is able to make from its position outside the security domain to be protected, and a more active attacker, that may have planted a Trojan Horse in the domain to be protected, and which seeks to use covert channels to pass information out of this domain. While this distinction turns out not to matter in asynchronous systems [18], in synchronous settings, it leads to two different definitions of security, known as Nondeducibility on Inputs (NDI) [41], and Nondeducibility on Strategies (NDS) [46]. (The term strategies in the latter refers to the strategies that a Trojan Horse may employ to pass information out of the security domain.) Considerations of proof methods for security, and compositionality of these methods, have led to the introduction of further definitions of security, such as *unwinding relations* [24] and the associated definition of *restrictiveness* (RES) [34].

* Corresponding author at: Oracle Labs, Brisbane, Australia.
E-mail address: chenyi.x.zhang@oracle.com (C. Zhang).
[1] Part of this work was performed while the author was at NICTA, Sydney, Australia.

http://dx.doi.org/10.1016/j.tcs.2016.03.011
0304-3975/

One of the dimensions along which it makes sense to evaluate a definition of security is the practicality of verification techniques it enables. The early literature on the topic was motivated primarily by theorem proving verification methods, but in recent years the feasibility of automated verification techniques has begun to be investigated [17–20,26,43]. This recent work on automated verification of security has dealt primarily with asynchronous systems models.

Our contribution In this paper we investigate the complexity of automated verification for a range of definitions of information flow in a *synchronous* systems model, in which agents are aware of a global clock and may use timing information in their deductions. This model is significant in that a number of timing-based covert channels been demonstrated [47,36] in modern processor designs, that leak information from one process to another with which it is not permitted to communicate. When one of the processes is performing a cryptographic computation, this could lead the other to learn information about cryptographic keys in use [27]. It is therefore desirable that systems designs are free of timing-based covert channels. The asynchronous definitions of security that have been the focus of much of the literature fail to ensure this.

We study three definitions of security in this paper: synchronous versions of Nondeducibility on Inputs (NDI), Nondeducibility on Strategies (NDS) and an unwinding based definition called Restrictiveness (RES). We consider just a two-agent setting, with agents L for a low security domain and H for a high security domain, and the (classical) security policy that permits H to know about L's activity, but prohibits L from knowing about the activity of H. We show that all three definitions are decidable in finite state systems, and with complexities of PSPACE-complete for NDI, EXPSPACE-complete for NDS, and polynomial time for RES. A preliminary version of this paper, with only proof sketches, appeared in [14]. In this extended version, we provide detailed proofs for all results.

Outline of the paper The structure of the paper is as follows. Section 2 introduces our systems model and the three definitions of security that we study. The following sections discuss the complexity results for each of these definitions. Section 3 deals with Nondeducibility on Inputs, Section 4 deals with Nondeducibility on Strategies, and Section 5 deals with the unwinding-based definition. Related literature is discussed in Section 6, and Section 7 makes some concluding remarks.

2. Semantic model and information flow security policies

2.1. Notation

Otherwise stated, we use standard notation from automata theory. Given a finite set (alphabet) A, we write A^* for the set of finite words over A. We denote the empty word by ϵ, and for $w \in A^*$, we write $|w|$ for the length of w. For $n \in \mathbb{N}$, A^n stands for the set of words of length n over A.

2.2. Synchronous machines

We work with a synchronous, non-deterministic state machine model for two agents, H and L. At each step of the computation, the agents (simultaneously) perform an action, which is resolved non-deterministically into a state transition. Both agents make (possibly incomplete) observations of the state of the system, and do so with awareness of the time. Time is discrete and measured by the number of steps in a computation.

Our machine model is given in the following definition. We do not make any finiteness assumptions in this section and the results in this section hold for this unconstrained model.

Definition 2.1 *(Synchronous machine).* A *synchronous machine* M is a tuple of the form $\langle S, A, s_0, \rightarrow, O, obs\rangle$ where

- S is the set of states,
- $A = A_H \times A_L$ is a set of joint actions (or joint inputs), each composed of an action of H from the set A_H and an action of L from the set A_L,
- s_0 is the initial state,
- $\rightarrow \subseteq S \times A \times S$ defines state transitions resulting from the joint actions,
- O is a set of observations,
- $obs : S \times \{H, L\} \rightarrow O$ represents the observations made by each agent in each state.

We write obs_u for the mapping $obs(\cdot, u) : S \rightarrow O$, and $s \xrightarrow{a} s'$ for $\langle s, a, s'\rangle \in \rightarrow$. We assume that machines are *input-enabled*, by requiring that for all $s \in S$ and $a \in A$, there exists $s' \in S$ such that $s \xrightarrow{a} s'$. We write $\mathbb{M}^s$ for the set of synchronous machines.

A *run* r of M is a finite sequence $r = s_0 a_1 s_1 \dots a_n s_n$ with: $a_i \in A$ and $s_i \xrightarrow{a_{i+1}} s_{i+1}$ for all $i = 0 \dots n-1$. We write $\mathcal{R}(M)$ for the set of all runs of M. We denote the sequence of joint actions $a_1 \dots a_n$ in the run r by $Act(r)$. For each agent $u \in \{H, L\}$ we define $proj_u : A \rightarrow A_u$ to be the projection of joint actions onto agent u's actions. We write $Act_u(r)$ for the sequence of agent u's actions in $Act(r)$, e.g., if $Act(r) = a_1 \dots a_n$ then $Act_u(r) = proj_u(a_1) \dots proj_u(a_n)$.

2.3. Agent views

For a sequence w, and $1 \leq i \leq |w|$, we write w_i for the i-th element of w, and $w[i]$ for the prefix of w up to the i-th element. We assume agents have a *synchronous* view of the machine, making an observation at each moment of time and being aware of each of their own actions (but not the actions of the other agent, which are given simultaneously and independently). Given a synchronous machine M, and $u \in \{H, L\}$, we define *u views* by the mapping $\mathtt{view}_u : \mathcal{R}(M) \to O(A_u O)^*$ by:

$$\mathtt{view}_u(s_0 a_1 s_1 a_2 \cdots a_n s_n) = obs_u(s_0)\, proj_u(a_1)\, obs_u(s_1)\, proj_u(a_2) \cdots proj_u(a_n)\, obs_u(s_n).$$

Intuitively, this says that an agent's view of a run is the history of all its state observations as well as its own actions in the run. We say that a sequence v of observations and actions is a *possible u view in a system M* if there exists a run r of M such that $v = \mathtt{view}_u(r)$. The mapping $\mathtt{view}_u$ extends straightforwardly to sets of runs $R \subseteq \mathcal{R}(M)$, by $\mathtt{view}_u(R) = \{\mathtt{view}_u(r) \mid r \in R\}$. We define the length $|v|$ of a view v to be the number of actions it contains.

2.4. Expressiveness issues

We remark that the model is sufficiently expressive to represent an alternate model in which agents act in turn under the control of a scheduler. We say that a synchronous machine is *scheduled* if for each state $s \in S$ either

- for all actions $a \in A_H$ and $b, b' \in A_L$, and states $t \in S$, $s \xrightarrow{(a,b)} t$ iff $s \xrightarrow{(a,b')} t$, or
- for all actions $a, a' \in A_H$ and $b \in A_L$, and states $t \in S$, $s \xrightarrow{(a,b)} t$ iff $s \xrightarrow{(a',b)} t$.

This definition says that state transitions in a scheduled machine are determined by the actions of *at most one* of the agents (the agent scheduled at that state); the other agent has no control over the transition. The model involving machines under the control of a scheduler of [44], in which at most one agent acts at each step of the computation, can be encoded as scheduled synchronous machines.

2.5. Notions of information flow security

We consider a number of different notions of information flow security. Each definition provides an interpretation for the security policy $L \to H$, which states that information is permitted to flow from L to H, but not from H to L. Our definitions are intended for synchronous systems, in which the agents share a clock and are able to make deductions based on the time. (Much of the prior literature has concentrated on asynchronous systems, in which an agent may not know how many actions another agent has performed.)

2.5.1. Non-deducibility on inputs

The first definition we consider states that L should not be able to infer H actions from its view.

Definition 2.2. A synchronous machine M satisfies *Non-Deducibility on Inputs* ($M \in \mathtt{NDI}$) if for every possible L view v in M and every sequence of H actions $\alpha \in A_H^{|v|}$, there exists a run $r \in \mathcal{R}(M)$ such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$.

Intuitively, in a synchronous system, L always knows how many actions H has performed, since this is always identical to the number of actions that L has itself performed. In particular, if L has made view v, then L knows that H has performed $|v|$ actions. The definition says that the system is secure if this is *all* that L can learn about what sequence of actions H has performed. Whatever L observes is consistent with any sequence of actions by H of this length.[2] More precisely, define $K_L(v)$ for an L view v to be the set of H action sequences $Act_H(r)$ for r a run with $v = \mathtt{view}_L(r)$; this represents what L knows about H's actions in the run. Then $M \in \mathtt{NDI}$ iff for all possible L views v we have $K_L(v) = A_H^{|v|}$.

The definition of NDI takes the viewpoint that a system is secure if it is not possible for L to make any nontrivial deductions about H behaviour, provided that H does not actively seek to communicate information to L. This is an appropriate definition when H is trusted not to deliberately act so as to communicate information to L, and the context is one where H is equally likely to engage in any of its possible behaviours. In some circumstances, however, NDI proves to be too weak a notion of security. In particular, this is the case if the attack model against which the system must be secure includes the possibility of Trojan Horses at the H end of the system, which must be prevented from communicating H secrets to L. The following example, due in essence to Wittbold and Johnson [46] shows that it is possible for a system to satisfy NDI, but still allow for L to deduce H information.

[2] Recall that M is input-enabled.

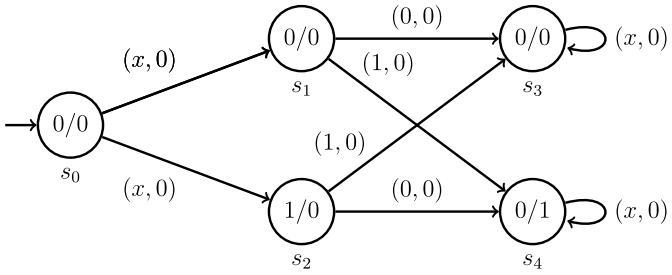

Fig. 1. A synchronous machine in NDI, but not in NDS, where $x \in \{0,1\}$.

Example 1. We present a synchronous machine that satisfies NDI in Fig. 1. We use the convention in such figures that the observations are shown on a state s in the form of $obs_H(s)/obs_L(s)$. Edges are labelled with joint actions (a, a') where $a \in A_H$ and $a' \in A_L$. When a is x this means that there is such an edge for all $a \in A_H$. In this example the action sets are $A_H = \{0,1\}$, $A_L = \{0\}$. Note that in state s_1 and s_2, L's observation in the next state is determined as the *exclusive-or* of H's current observation and H's action. The system is in NDI since every H action sequence is compatible with every L view of the same length. For example, the L view 00000 is consistent with H action sequence 00 and 10 (path $s_0s_1s_3$) and with H action sequence 01 and 11 (path $s_0s_2s_3$). Nevertheless, H can communicate a bit b of information to L, as follows. Note that H is able to distinguish between state s_1 and s_2 by means of the observation it makes on these states (at time 1). Suppose $b = 1$, then H chooses action 1 at s_1 and action 0 at s_2; in either case the next state is s_4, and L observes 1. Alternately, if $b = 0$, then H chooses action 0 at s_1 and action 1 at s_2; in either case the next state is s_3, and L observes 0. Whatever the value of b, H has guaranteed that L observes b at time 2, so this bit has been communicated. Intuitively, this means that the system fails to block Trojan Horses at H from communicating with L, even though it satisfies NDI. (The structure can be repeated so that H can communicate a message of any length to L in plain text.) □

2.5.2. Non-deducibility on strategies

The essence of Example 1 is that L is able to deduce H secrets based not just on its knowledge of the system, but also its knowledge that H is following a particular strategy for communication of information to L. In response to this example, Wittbold and Johnson proposed the following stronger definition of security that they called *non-deducibility on strategies*. To state this definition, we first formalize the possible communication strategies that can be used by H. Intuitively, H's behaviour may depend on what H has been able to observe in the system.

Definition 2.3 *(H strategy, consistent runs).* An *H strategy* in M is a function $\pi : \mathtt{view}_H(\mathcal{R}(M)) \to A_H$ mapping each possible view of H (in M) to an H action. A run $r = s_0a_1s_1 \ldots a_ns_n$ of M is *consistent with* an H strategy π if for all $i = 0 \ldots n-1$, we have $proj_H(a_{i+1}) = \pi(\mathtt{view}_H(s_0a_1s_1 \ldots a_is_i))$. We write $\mathcal{R}(M,\pi)$ for the set of runs of M that are consistent with the H strategy π.

We can now state Wittbold and Johnson's definition.

Definition 2.4. A synchronous system M satisfies *Nondeducibility on Strategies* ($M \in$ NDS), if for all H strategies π_1, π_2 in M, we have $\mathtt{view}_L(\mathcal{R}(M,\pi_1)) = \mathtt{view}_L(\mathcal{R}(M,\pi_2))$.

Intuitively, this definition says that the system is secure if L is not able to distinguish between different H strategies by means of its views. In Example 1, given an H strategy π_1 satisfying $\pi_1(0x0) = 0$ and $\pi_1(0x1) = 1$, and another H strategy π_2 satisfying $\pi_2(0x0) = 1$ and $\pi_2(0x1) = 0$, we have the L view 00001 in $\mathtt{view}_L(\mathcal{R}(M,\pi_2)$ but not in $\mathtt{view}_L(\mathcal{R}(M,\pi_1))$. Thus, the sets of L views differ for these two strategies, so the system is not in NDS.

An alternate formulation of the definition can be obtained by noting that for every possible L view v, there is an H strategy π such that $v \in \mathtt{view}_L(\mathcal{R}(M,\pi))$, viz., if $v = \mathtt{view}_L(r)$, we take π to be a strategy that always performs the same action at each time $i < |r|$ as H performs at time i in r. Thus, we can state the definition as follows:

Proposition 1. *$M \in$ NDS iff for all H strategies π in M, we have $\mathtt{view}_L(\mathcal{R}(M,\pi)) = \mathtt{view}_L(\mathcal{R}(M))$.*

Proof. It is trivial that if $\mathtt{view}_L(\mathcal{R}(M,\pi)) = \mathtt{view}_L(\mathcal{R}(M))$ for all strategies π then $M \in$ NDS. Conversely, suppose that $M \in$ NDS, and let π be any strategy. Plainly $\mathtt{view}_L(\mathcal{R}(M,\pi)) \subseteq \mathtt{view}_L(\mathcal{R}(M))$; we show the reverse containment. Let $v \in \mathtt{view}_L(\mathcal{R}(M))$ be a possible L view. By the above observation there exists a strategy π_1 such that $v \in \mathtt{view}_L(\mathcal{R}(M,\pi_1))$. By $M \in$ NDS, $\mathtt{view}_L(\mathcal{R}(M,\pi_1)) = \mathtt{view}_L(\mathcal{R}(M,\pi))$, so also $v \in \mathtt{view}_L(\mathcal{R}(M,\pi))$, as required. □

This formulation makes it clear that H cannot communicate any information to L by means of its strategies. It is also apparent that allowing H strategies to be non-deterministic (i.e., functions from H views to a *set* of H actions) would not

lead to a different definition of NDS, since the more choices H has in a strategy the more L-views are compatible with that strategy. We remark that in asynchronous systems (in which we use an asynchronous notion of view), similarly defined notions of non-deducibility on inputs and non-deducibility on strategies turn out to be equivalent [18,42]. The Example 1 above shows that this is not the case in synchronous machines, where the two notions are distinct.

2.5.3. Unwinding relations

Nondeducibility-based definitions of security are quite intuitive, but they turn out to have some disadvantages as a basis for secure systems development. In particular, non-deducibility on inputs is not compositional: combining two systems, each secure, can produce a compound system that is not secure [34]. For this reason, some stronger, but less intuitive definitions have been advocated in the literature.

One of these, McCullough's notion of restrictiveness [34], is closely related to an approach to formal proof of systems security based on what are known as "unwinding relations." A variety of definitions of unwinding relations have been proposed in the literature [24,38,31,7], in the context of a number of different underlying systems models and associated definitions of security for which they are intended to provide a proof technique. We propose here a variant of such definitions that is appropriate to the machine model we consider in this paper, drawing on definitions proposed by van der Meyden and Zhang [44] for machines acting under the control of a scheduler.

Definition 2.5 *(Synchronous unwinding relation).* A *synchronous unwinding relation* on a system M is a symmetric relation $\sim \subseteq S \times S$ satisfying the following:

1. $s_0 \sim s_0$,
2. $s \sim t$ implies $obs_L(s) = obs_L(t)$, and
3. $s \sim t$ implies that for all $a_1, a_2 \in A_H$ and $a_3 \in A_L$, if $s \xrightarrow{(a_1,a_3)} s'$ then there exists a state t' such that $t \xrightarrow{(a_2,a_3)} t'$, and $s' \sim t'$.

Intuitively, an unwinding relation is a bisimulation-like relation over S that shows L observations are locally uncorrelated with H actions.

Definition 2.6. A synchronous machine M satisfies restrictiveness ($M \in$ RES), if there exists a synchronous unwinding relation on M.

Part of the significance of RES is that it provides a proof technique for our notions of nondeducibility, as shown by the following result, which relates the three notions of security we have introduced:

Theorem 1. *The following containments hold and are strict:* RES $\subset$ NDS $\subset$ NDI.

Proof. To show that RES $\subseteq$ NDS we argue as follows. Suppose that $\sim$ is a synchronous unwinding on M. Let v be any possible L view, and π any H strategy. We have to show that $v \in \mathtt{view}_L(\mathcal{R}(M,\pi))$. For this, let $r = s_0a_1s_1 \ldots a_ns_n$ be a run such that $v = \mathtt{view}_L(r)$. We show that there exists a run $r' = s'_0a'_1s'_1 \ldots a'_ns'_n$ consistent with π such that $v = \mathtt{view}_L(r')$. We proceed inductively, showing for each $i = 0 \ldots n$ that $\mathtt{view}_L(s_0a_1s_1 \ldots a_is_i) = \mathtt{view}_L(s'_0a'_1s'_1 \ldots a'_is'_i)$ and $s_i \sim s'_i$, where $s'_0a'_1s'_1 \ldots a'_is'_i$ is a run consistent with π. In the base case, we have $s'_0 = s_0$ and the claim is trivial. For the inductive case, let $a = \pi(\mathtt{view}_H(s'_0a'_1s'_1 \ldots a'_is'_i))$ and $b = proj_L(a_{i+1})$, and take $a'_{i+1} = (a, b)$. Since $proj_L(a_{i+1}) = proj_L(a'_{i+1})$ and $\sim$ is an unwinding relation, there exists a state s'_{i+1} such that $s'_i \xrightarrow{a'_{i+1}} s'_{i+1}$ and $s_{i+1} \sim s'_{i+1}$. Further, we conclude that $obs_L(s_{i+1}) = obs_L(s'_{i+1})$. Since $s'_0a'_1s'_1 \ldots a'_is'_i$ is consistent with π, so is $s'_0a'_1s'_1 \ldots a'_is'_ia'_{i+1}s'_{i+1}$, and

$$\begin{aligned}\mathtt{view}_L(s'_0a'_1s'_1 \ldots a'_is'_ia'_{i+1}s'_{i+1}) &= \mathtt{view}_L(s'_0a'_1s'_1 \ldots a'_is'_i)proj_L(a'_{i+1})obs_L(s'_{i+1}) \\ &= \mathtt{view}_L(s_0a_1s_1 \ldots a_is_i)proj_L(a_{i+1})obs_L(s_{i+1}) \\ &= \mathtt{view}_L(s_0a_1s_1 \ldots a_is_ia_{i+1}s_{i+1}),\end{aligned}$$

as required.

Next we show that NDS $\subseteq$ NDI. Let $\alpha \in A_H^*$ and v be an L observation satisfying $|\alpha| = |v|$. We construct a "blind" H strategy π as $\pi(v') = \alpha_{|v'|+1}$ if $|v'| < |\alpha|$ and $\pi(v') = a_H$ otherwise, where a_H is an arbitrary action in A_H. Since $M \in$ NDS, we have $\mathtt{view}_L(\mathcal{R}(M,\pi)) = \mathtt{view}_L(\mathcal{R}(M))$, so there exists a run $r \in \mathcal{R}(M,\pi)$ such that $\mathtt{view}_L(r) = v$. By the construction of π we have $Act_H(r) = \alpha$.

That the inclusions are strict follows from Example 1 and Example 2 below. □

Example 2. We present a machine in Fig. 2 that satisfies NDS but does not satisfy RES. In this system we let $A_H = \{0, 1\}$, $A_L = \{0\}$. We use the conventions from Example 1. One may easily observe that the set of L views is given by the regular

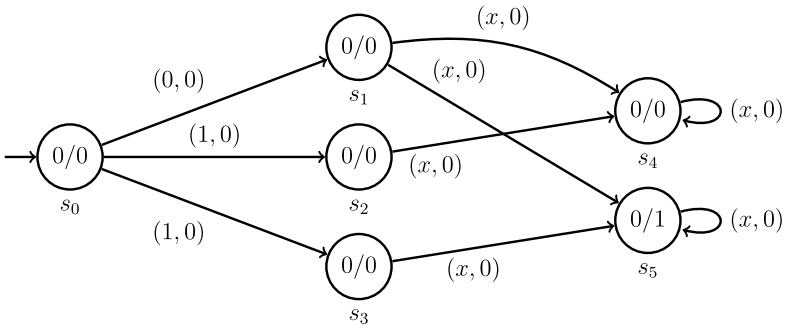

Fig. 2. A synchronous machine M in NDS, but not in RES, where $x \in \{0,1\}$. Every state s is labelled with a pair $obs_H(s)/obs_L(s)$.

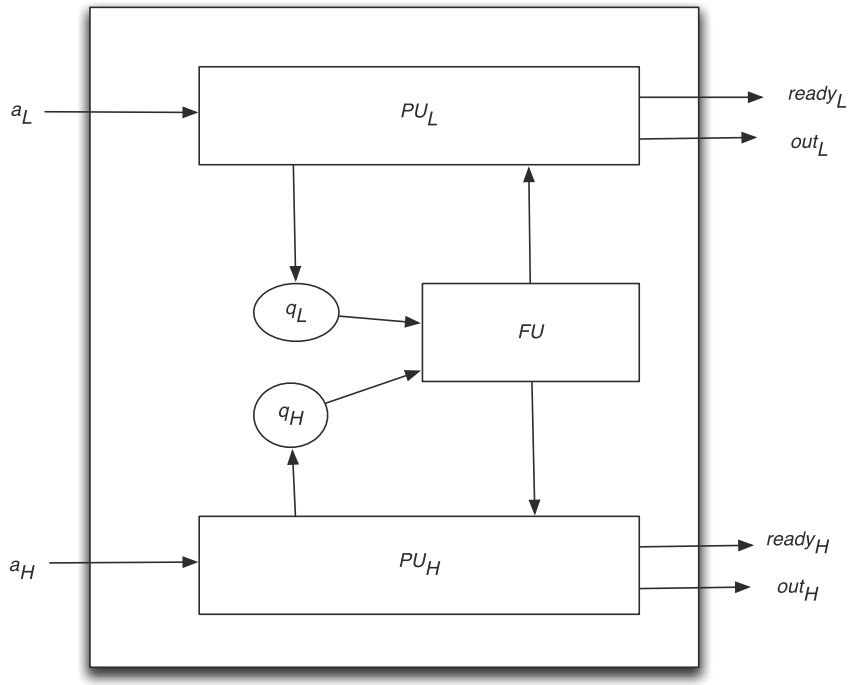

Fig. 3. An SMT processor with shared functional unit FU.

language $000((00)^* + (01)^*)$ and all the views are compatible with every possible H strategy. However, there does not exist a synchronous unwinding relation. Suppose there was such a relation $\sim$. Then $s_0 \sim s_0$, and for joint actions $(0,0)$ and $(1,0)$, we have $s_0 \xrightarrow{(0,0)} s_1$, $s_0 \xrightarrow{(1,0)} s_2$ and $s_0 \xrightarrow{(1,0)} s_3$, and we would require s_1 to be related to either s_2 or s_3. However, neither s_2 nor s_3 can be related to s_1: from s_2 user L can only observe $(00)^*$ in the future, and from s_3 only $(01)^*$ can be observed by L. Note from s_1 both $(00)^*$ and $(01)^*$ are possible for L. □

In the following sections, we study the complexity of the notions of security we have defined above.

2.6. Example

To illustrate the synchronous models and definitions we study in this paper in a more realistic setting, we sketch an example that shows how covert channels can arise from resource sharing in hardware designs. In particular, the following is illustrative of covert channels found in simultaneous multi-threaded processors [47]. Such a processor is able to fetch instructions and produce outputs simultaneously (i.e., within the same clock cycle) from two parallel processes or threads. Consider a device (depicted in Fig. 3) that reads instructions from two input sources, H and L. The instructions are of two types: easy (e.g. addition) and hard (e.g., multiplication) and we have the decomposition $A_u = Easy_u \cup Hard_u$ for $u \in \{L, H\}$. For each security level u, there are two outputs: $ready_u$ and out_u. Intuitively, out_u is the result output to the user, and $ready_u$ is a flag that indicates that an output is available.

Internally, there is an independent processing unit PU_u for each $u \in \{L, H\}$ to process the easy instructions, but for processing the hard instructions, H and L share the use of a special functional unit FU. Two queues, q_L, q_H store inputs for the functional unit. We assume here for the sake of simplicity that the queues have maximal length 1, but similar

conclusions about security would hold for any finite queue length. The functional unit maintains a variable *turn* to capture the currently scheduled input source.

The processing units PU_u operate as follows in each clock cycle. If q_u is not empty, then there is a pending computation and input a_u is ignored. Otherwise, the input is read. If $a_u \in Easy_u$ then the results of operation a_u are computed by PU_u, the result is stored in out_u, and flag $ready_u$ is set to 1. If $a_u \in Hard_u$ then a_u is stored in the queue q_u and $ready_u$ is set to 0 to indicate that the output is not yet available.

The functional unit *FU* operates as follows. First, it selects one of the agents u, according to some scheduling policy: we discuss two variants of this below. If queue q_u is empty then *FU* does nothing in the current round (except for operations necessary to maintain its scheduler state.) Otherwise, the operation a_u in queue q_u is retrieved and the queue emptied, the operation is performed, and the result stored in out_u, and flag $ready_u$ is set to 1. Finally, *turn* is updated, according to the scheduling policy. Note that both *FU* and PU_u may write to u's output variables, but there is no actual conflict.

The security of such a system depends on the scheduling policy used by *FU*. Suppose first that, in the interests of fairness and throughput, after FU has handled an agent's instruction, it prioritizes the opposite agent in the event of contention. Initially the priority is set to H (i.e., $turn = H$ at start-up). If q_{turn} is nonempty, then its operation is selected for handling, and *turn* is flipped. Otherwise, any operation in the other queue is selected for handling, but *turn* is not changed (the same agent still has priority). Intuitively, this setting is not secure, and the system can be shown to be not in NDI. Suppose that L inputs a sequence a_1a_2 of actions where a_1 is an instruction in $Hard_L$. Simultaneously, H inputs the sequence of actions b_1b_2. After a_1, L will observe $ready_L = 0$, since the first hard instruction is always placed in the queue q_L. However, after a_2, L will observe either $ready_L = 0$ or $ready_L = 1$, depending on the action b_1. The observation $ready_L = 0$ will occur in case $b_1 \in Hard_H$: this operation is placed into q_H in the first step and selected for processing by *FU* in the second step (delaying the processing of a_1). Alternately, if $b_1 \in Easy_H$, then q_H is empty after the first step, enabling *FU* to select the action a_1 in q_L for processing, giving Low observation $ready_L = 1$ after the second step. Agent L can therefore deduce information about agent H's actions, meaning that the system does not satisfy NDI. It can be seen that repeated use of this flow of information gives a high throughput covert channel for messages to be passed from H to L. For more concrete and realistic examples (e.g., written in the C programming language or machine code), we refer to the literature [36,47].

Alternately, consider a simple alternating scheduler for *FU*, in which always selects an action, if there is one, from q_H at even times, and always selections an action, if there is one, from q_L at odd times, but does not select an action in the other queue should the preferred queue be empty. With this scheduler, there is, intuitively, no correlation between L observations and H actions, and the system is secure. Indeed, it can be shown to satisfy the strongest of our security definitions RES. We leave the details to the reader.

3. Synchronous nondeducibility on inputs

In this section we establish the following result:

Theorem 2. *For the class of finite state synchronous machines,* NDI *is* PSPACE*-complete with respect to logspace reductions.*

3.1. PSPACE-easiness

Stating the definition in the negative, a system is not in NDI if there exists an L view v and a sequence of H actions α with $|\alpha| = |v|$ such that there exists no run r with $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$. We show that NDI is decidable by a procedure that searches for such an L view v and H action sequence α. The key element of the proof is to show that we need to maintain only a limited amount of information during this search, so that we can bound the length of the witness (v, α), and the amount of space needed to show that such a witness exists.

To show this, suppose we are given a machine $M = \langle S, A, s_0, \rightarrow, O, obs\rangle$. Given a sequence $\alpha \in A_H^*$ and a sequence $v \in O(A_L O)^*$, we define the set $K(\alpha, v)$ to be the set of all final states of runs r of M consistent with α and v, i.e., such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$. For each $a \in A_H$, $b \in A_L$ and $o \in O$, we also define the function $\delta_{a,b,o} : \mathcal{P}(S) \to \mathcal{P}(S)$, by

$$\delta_{a,b,o}(T) = \{t \in S \mid \text{for some } t' \in T \text{ we have } t' \xrightarrow{(a,b)} t \text{ and } obs_L(t) = o\}\ .$$

For the system M define the labelled transition system $LTS(M) = (Q, \Sigma, q_0, \Rightarrow)$ as follows:

1. $Q = S \times \mathcal{P}(S)$,
2. $q_0 = (s_0, \{s_0\})$,
3. $\Sigma = A_H \times A_L \times A_H$,
4. $\Rightarrow \subseteq Q \times \Sigma \times Q$ is the labelled transition relation defined by $(s, T) \Rightarrow^{(a,b,a')} (s', T')$ if $a \in A_H$, $b \in A_L$, $a' \in A_H$ such that $s \xrightarrow{(a,b)} s'$ and $T' = \delta_{a',b,obs_L(s')}(T)$.

Intuitively, the component s in a state $(s, T) \in Q$ is used to ensure that we generate an L view v that is in fact possible. The components a, b in a transition $(s, T) \Rightarrow^{(a,b,a')} (s', T')$ represent the actions used to generate the run underlying v, and

the component a' is used to generate a sequence α. The set T represents $K(\alpha, v)$. More precisely, we have the following result:

Lemma 1. *If $q_0 \Rightarrow^{(a_1,b_1,a'_1)} (s_1, T_1) \Rightarrow \cdots \Rightarrow^{(a_n,b_n,a'_n)} (s_n, T_n)$, then the sequence $v = obs_L(s_0)b_1obs_L(s_1)\ldots b_nobs_L(s_n)$ is a possible L view, and $\alpha = a'_1 \ldots a'_n$ is a sequence of H actions such that $|v| = |\alpha|$ and $K(\alpha, v) = T_n$.*

Conversely, for every possible L view v with $|v| = n$, and sequence of H actions $\alpha = a'_1 \ldots a'_n$, there exists a path $q_0 \Rightarrow^{(a_1,b_1,a'_1)} (s_1, T_1) \Rightarrow \cdots \Rightarrow^{(a_n,b_n,a'_n)} (s_n, T_n)$ such that $v = obs_L(s_0)b_1obs_L(s_1)\ldots b_nobs_L(s_n)$ and $K(\alpha, v) = T_n$.

Proof. We first show that for all $\alpha \in A_H^*$, $v \in O(A_L O)^*$, $a \in A_H$, $b \in A_L$ and $o \in O$, we have $K(\alpha a, vbo) = \delta_{a,b,o}(K(\alpha, v))$. To show $K(\alpha a, vbo) \subseteq \delta_{a,b,o}(K(\alpha, v))$, suppose that $t \in K(\alpha a, vbo)$. Then there exists a run r of M such that $Act_H(r) = \alpha a$ and $\mathtt{view}_L(r) = vbo$ and the final state of r is t. It follows that $obs_L(t) = o$. Thus, we may write $r = r' \xrightarrow{(a,b)} t$, where $Act_H(r') = \alpha$, and $\mathtt{view}_L(r') = v$. Thus, the final state t' of r' is in $K(\alpha, v)$. Since $t' \xrightarrow{(a,b)} t$ and $obs_L(t) = o$, it follows that $t \in \delta_{a,b,o}(K(\alpha, v))$.

Conversely, if $t \in \delta_{a,b,o}(K(\alpha, v))$ then by definition of $\delta_{a,b,o}$ there exists $t' \in K(\alpha, v)$ such that $t' \xrightarrow{(a,b)} t$ and $obs_L(t) = o$. By definition of $K(\alpha, v)$ there exists a run r of M such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$. Taking $r' = r \xrightarrow{(a,b)} t$, we see that r' is a run of M with $Act_H(r) = \alpha a$ and $\mathtt{view}_L(r') = vbo$. Thus, $t \in K(\alpha a, vbo)$, as required. This completes the proof that $K(\alpha a, vbo) = \delta_{a,b,o}(K(\alpha, v))$.

We can now prove the two parts of the result:

- Suppose $q_0 \Rightarrow^{(a_1,b_1,a'_1)} (s_1, T_1) \Rightarrow \cdots \Rightarrow^{(a_n,b_n,a'_n)} (s_n, T_n)$ is a run of $LTS(M)$, then by definition $r = s_0 \xrightarrow{(a_1,b_1)} s_1 \xrightarrow{(a_2,b_2)} \ldots \xrightarrow{(a_n,b_n)} s_n$ is a run of M, such that $\mathtt{view}_L(r) = obs_L(s_0)b_1obs_L(s_1)\ldots b_nobs_L(s_n) = v$ is a possible L view. Moreover, $T_{i+1} = \delta_{a_i,b_i,obs_L(s_{i+1})}(T_i)$ for $i = 0 \ldots n-1$, where we take $T_0 = \{s_0\}$. Since $T_0 = K(\varepsilon, obs_L(s_0))$, it follows from the above using a straightforward induction that $T_n = K(\alpha, v)$, where $\alpha = a'_1 \ldots a'_n$.
- Let v be a possible L view, then there exists a run $r = s_0 \xrightarrow{(a_1,b_1)} s_1 \xrightarrow{(a_2,b_2)} \ldots \xrightarrow{(a_n,b_n)} s_n$ of M such that $\mathtt{view}_L(r) = v$. Given a sequence of H actions $\alpha = a'_1 \ldots a'_n$, we inductively define $T_0 = \{s_0\}$ and $T_{i+1} = \delta_{a_i,b_i,obs_L(s_{i+1})}(T_i)$. It is then immediate by definition that we have a path $(s_0, \{s_0\}) \Rightarrow^{(a_1,b_1,a'_1)} (s_1, T_1) \Rightarrow \cdots \Rightarrow^{(a_n,b_n,a'_n)} (s_n, T_n)$ in $LTS(M)$. By a straightforward induction using what was proved above, we have that $T_n = K(\alpha, v)$, where $\alpha = a'_1 \ldots a'_n$ and $v = obs_L(s_0)b_1obs_L(s_1)\ldots b_nobs_L(s_n)$. □

We now note that for an H action sequence α and a possible L view v, with $|v| = |\alpha|$, there exists no run r such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$ iff $K(\alpha, v) = \varnothing$. The existence of such a pair (α, v), is therefore equivalent, by Lemma 1, to the existence of a path in $LTS(M)$ from q_0 to a state (s, T) with $T = \varnothing$. This can be decided in NSPACE$(O(|M|)) =$ DSPACE$(O(|M|^2)) \subseteq$ PSPACE. This proves the following theorem.

Theorem 3. *$M \in$ NDI is decidable in* PSPACE.

We note, moreover, that since there are at most $|S| \times 2^{|S|}$ states in Q, if there exists a pair (α, v) witnessing that $M \notin$ NDI there exists such a pair with $|\alpha| \leq |S| \times 2^{|S|}$.

3.2. PSPACE-hardness

We show that NDI is PSPACE-hard already in the special case of scheduled machines. The proof is by a polynomial time reduction from the problem of deciding, given a non-deterministic finite state automaton $\mathcal{A}$ on alphabet Σ, if the language $L(\mathcal{A})$ accepted by $\mathcal{A}$ is equal to Σ^*. This *Universality* problem is PSPACE-hard [40].

Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \delta, F\rangle$ be a non-deterministic finite state automaton (without ε-transitions), with states Q, initial states $Q_0 \subseteq Q$, alphabet Σ, transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, and final states F. We construct a machine $M(\mathcal{A})$, depicted in Fig. 4, that is in NDI iff $L(\mathcal{A}) = \Sigma^*$. Low actions in this machine correspond to letters $a \in \Sigma$, and H has two actions h, h'. Intuitively, the machine consists of two components. One simulates the automaton using L actions as input letters, but can only be entered if the first action of H is h. Acceptance of the input word is represented in this component by L making observation 1. The other generates all possible L views, and can be entered if the first action of H is h'. If any word is not in $L(\mathcal{A})$, then the L view that corresponds to this word can only be obtained when the first H input is h', so the system is not in NDI.

Formally, we define $M(\mathcal{A}) = \langle S, A, s_0, \to, obs, O\rangle$ to be a scheduled machine, and use a function $sched : S \to \{H, L\}$ to indicate the agent (if any) whose actions determine transitions. In view of this, when $sched(s) = u$ and $a \in A_u$, we may write $s \xrightarrow{a} t$ to represent that $s \xrightarrow{b} t$ for all joint actions b with $proj_u(b) = a$. The components of $M(A)$ are defined as follows.

- $S = Q \cup \{s_0, s_1, s_2, s_3\}$, where $Q \cap \{s_0, s_1, s_2, s_3\} = \varnothing$,

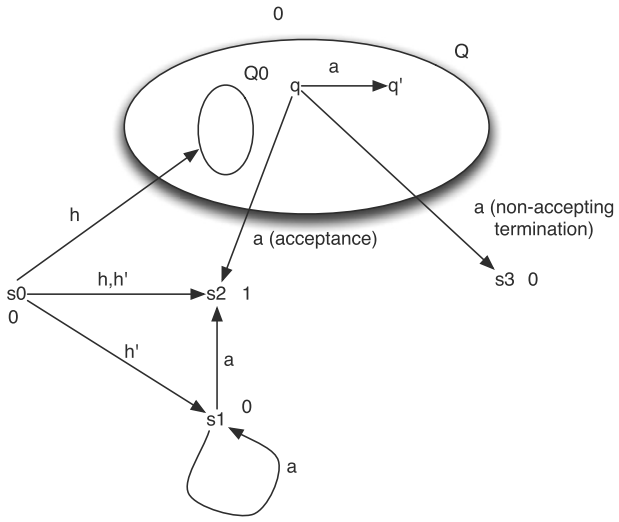

Fig. 4. Construction for PSPACE hardness of NDI.

- $sched(s_0) = H$ and $sched(s) = L$ for all $s \in S \setminus \{s_0\}$,
- $A = A_H \cup A_L$ where $A_L = \Sigma$ and $A_H = \{h, h'\}$,
- $O = \{0, 1\}$,
- $obs : \{H, L\} \times S \to O$ with $obs_H(s) = 0$ for all $s \in S$ and $obs_L(s) = 0$ for all $s \in S \setminus \{s_2\}$, and $obs_L(s_2) = 1$.
- $\longrightarrow \subseteq S \times A \times S$ is defined as consisting of the following transitions (using the convention noted above)
  - $s_0 \xrightarrow{h} q$ for all $q \in Q_0$,
  - $s_0 \xrightarrow{h} s_2$, provided $Q_0 \cap F \neq \varnothing$,
  - $s_0 \xrightarrow{h'} s_1$ and $s_0 \xrightarrow{h'} s_2$,
  - $s_1 \xrightarrow{a} s_1$ and $s_1 \xrightarrow{a} s_2$ for all $a \in \Sigma$,
  - $s_2 \xrightarrow{a} s_2$ for all $a \in \Sigma$,
  - $s_3 \xrightarrow{a} s_3$ for all $a \in \Sigma$,
  - for $q, q' \in Q$ and $a \in A_L = \Sigma$ we have $q \xrightarrow{a} q'$ for all $q' \in \delta(q, a)$,
  - for $q \in Q$ and $a \in A_L = \Sigma$ such that $\delta(q, a) \cap F \neq \varnothing$, we have $q \xrightarrow{a} s_2$,
  - for $q \in Q$ and $a \in A_L = \Sigma$ such that $\delta(q, a) = \varnothing$, we have $q \xrightarrow{a} s_3$.

The construction of $M(\mathcal{A})$ from $\mathcal{A}$ can be done in logspace.

Intuitively, the runs of $M(\mathcal{A})$ produce two sets of L views, depending on whether the first H action is h or h'. In all circumstances, runs in which H does h' in the first step, with the first transition to s_1, produce an L view for each sequence in $0\Sigma 1(\Sigma 1)^*$ or $0\Sigma 0(\Sigma 0)^*(\Sigma 1)^*$ by switching from to s_2 at the first occurrence of observation 1. Runs in which H does h in the first step with a transition to a state in Q_0, correspond to simulations of $\mathcal{A}$ and produce two types of L views:

1. any sequence in $0\Sigma 0(\Sigma 0)^*$ (by means of a run that stays in Q for as long as possible, and moves to s_3 whenever an action is not enabled), and
2. any sequence of the form $0\Sigma 0 b_1 0 \dots b_{n-1} 0 b_n 1(\Sigma 1)^*$ with $b_1 \dots b_n \in L(\mathcal{A})$ (these come from runs that pass through Q and then jump to s_2).

In case $\varepsilon \in L(\mathcal{A})$, i.e., $Q_0 \cap F \neq \varnothing$, then also all sequences in $0\Sigma 1(\Sigma 1)^*$ are produced as the L view on a run in which the first transition, with H action h, is to s_2.

Note that since L is always scheduled after the first step, replacing any action by H after the first step in a run by any other action of H results in another run, with no change to the L view. Thus, the only thing that needs to be checked to determine whether $M(\mathcal{A}) \in \texttt{NDI}$ is whether the same can be said for the first step. Moreover, it can be seen from the above that, for all $\mathcal{A}$, and independently of whether $\varepsilon \in L(\mathcal{A})$, any L view obtained from a run in which the first H action is h can also be obtained from a run in which the first H action is h'. Thus, to show $M(\mathcal{A}) \in \texttt{NDI}$, it suffices to check that any L view obtained from a run in which the first H action is h' can also be obtained from a run in which the first action is h.

Proposition 2. $L(\mathcal{A}) = \Sigma^*$ *iff* $M(\mathcal{A}) \in \texttt{NDI}$.

Proof. For the 'only if' part, suppose $L(\mathcal{A}) = \Sigma^*$. We show that $M(\mathcal{A}) \in \texttt{NDI}$. As argued above, it suffices to show that any view obtained from a run in which the first H action is h' can also be obtained from a run in which the first action

is h. Let $r = s_0(h', b_1)t_1 \ldots (a_n, b_n)t_n$ be a run of $M(\mathcal{A})$, with the $a_i \in A_H$ and the $b_i \in A_L$. If $t_1 = s_2$ then since $\varepsilon \in L(\mathcal{A})$, simply by replacing the first transition by $s_0 \xrightarrow{(h,b_1)} s_2$ we obtain a run with first H action h that has exactly the same L view. Otherwise $t_1 = s_1$. If all $obs_L(t_i) = 0$ then we may construct a run of $M(\mathcal{A})$ with the same L view as r by taking any transition into Q in the first step, then remaining within Q throughout, or making a transition to s_3 if there is no enabled transition of $\mathcal{A}$ on the given input b_i. Otherwise, let i be the least index with $obs_L(t_i) = 1$. Since $L(\mathcal{A}) = \Sigma^*$, there exists a run $q_0 \xrightarrow{b_2} q_1 \ldots \xrightarrow{b_i} q_{i-1}$ of $\mathcal{A}$ with $q_{i-1} \in F$. By construction of $M(\mathcal{A})$, it then follows that

$$r' = s_0 \xrightarrow{(h,b_1)} q_0 \xrightarrow{(h,b_1)} q_1 \ldots q_{i-2} \xrightarrow{(a_i,b_i)} s_2 \ldots \xrightarrow{(a_n,b_n)} s_2$$

is a run with the same L view as r but with first H action h. Thus, $M(\mathcal{A}) \in \mathtt{NDI}$.

For the 'if' part, suppose there is a word $w = a_1 a_2 \ldots a_n \notin L(\mathcal{A})$. If $w = \varepsilon$, then the transition $s_0 \xrightarrow{h} s_2$ is not present in $M(\mathcal{A})$, so there is no run starting with H action h that produces the L view $0b1$ (with $b \in \Sigma$) that we get from $s_0 \xrightarrow{(h',b)} s_1$. Otherwise, for an arbitrary $a_0 \in \Sigma$, the L view $0a_0 0a_1 0a_2 \ldots a_n 1$ cannot be obtained from runs in which the first H action is h, because otherwise w would be accepted by $\mathcal{A}$. However this view is obtained from a run in which the first action is h'. Therefore $M(\mathcal{A}) \notin \mathtt{NDI}$. □

As pointed out at the beginning of this section, this lower bound result already holds for scheduled machines, and thus NDI is already PSPACE-hard for this subclass.

4. Nondeducibility on strategies

In this section we establish the following theorem:

Theorem 4. *For the class of finite state synchronous machines, and with respect to log-space reductions,* NDS *is* EXPSPACE*-complete.*

4.1. EXPSPACE-easiness

For the proof that NDS is decidable in EXPSPACE, we show that the problem is in DSPACE($2^{O(|M|)}$). It is convenient for this section to consider strategies π that are defined over the larger set $\mathcal{V}_H = O(A_H O)^*$ of candidate views of H, rather than the subset $\mathtt{view}_H(\mathcal{R}(M))$ of possible views.

We use the characterization of NDS given in Proposition 1. Let π be an H strategy, and β be an L view. Say that π *excludes* β if there does not exist a run r consistent with π such that $\beta = \mathtt{view}_L(r)$. Since always $\mathcal{R}(M, \pi) \subseteq \mathcal{R}(M)$, by Proposition 1, a system M satisfies NDS if and only if it is not the case that there exists a possible L view β in M and a strategy π such that π excludes β.

Our decidability result and complexity bound is obtained by showing that if such a strategy exists, then there is one of a particular normal form, and it can be found using a space-bounded search. The normal form strategies have a uniform structure, in that the choice of next action on an H view depends only on the length of the view and the set of states that H considers possible after that view, given that L's view β has not yet been excluded. We call this set of states H's *knowledge set*.

More precisely, the knowledge sets are defined as follows. Given a candidate H view $\alpha \in O(A_H O)^*$ and a (to be excluded) candidate L view $\beta \in O(A_L O)^*$ with $|\alpha| \leq |\beta|$, define $K(\alpha, \pi, \beta)$ to be the set of all final states of runs r consistent with π such that $\mathtt{view}_H(r) = \alpha$ and $\mathtt{view}_L(r)$ is a prefix of β.

These knowledge sets can be obtained in an incremental way using the update operators $\delta_{a_H,o_H,a_L,o_L} : \mathcal{P}(S) \to \mathcal{P}(S)$ defined for each $a_H \in A_H$, $a_L \in A_L$, and $o_H, o_L \in O$, to map $T \in \mathcal{P}(S)$ to

$$\delta_{a_H,o_H,a_L,o_L}(T) = \{s \in S \mid \exists t \in T \text{ with } t \xrightarrow{(a_H,a_L)} s \text{ and } obs_H(s) = o_H \text{ and } obs_L(s) = o_L\}.$$

The incremental characterisation is given in the following lemma.

Lemma 2. *Suppose that* $\pi(\alpha) = a_H$ *and* $|\alpha| = |\beta|$. *Then* $\delta_{a_H,o_H,a_L,o_L}(K(\alpha, \pi, \beta)) = K(\alpha a_H o_H, \pi, \beta a_L o_L)$.

Proof. We first show that

$$\delta_{a_H,o_H,a_L,o_L}(K(\alpha, \pi, \beta)) \subseteq K(\alpha a_H o_H, \pi, \beta a_L o_L).$$

Suppose $t \in \delta_{a_H,o_H,a_L,o_L}(K(\alpha, \pi, \beta))$. We show that $t \in K(\alpha a_H o_H, \pi, \beta a_L o_L)$. We have that there exists $s \in K(\alpha, \pi, \beta)$ such that $s \xrightarrow{(a_H,a_L)} t$ and $obs_H(t) = o_H$ and $obs_L(t) = o_L$. Thus there exists a run r, consistent with π, and with final state s, such that $\mathtt{view}_H(r) = \alpha$ and $\mathtt{view}_L(r) = \beta$. Since $\pi(\alpha) = a_H$, we obtain that the run $r(a_H, a_L)t$ is consistent with π and justifies $t \in K(\alpha a_H o_H, \pi, \beta a_L o_L)$.

Conversely, suppose $t \in K(\alpha a_H o_H, \pi, \beta a_L o_L)$. Then there exists a run r' consistent with π, which can be written in the form $r(a_H, a_L)t$ with $\mathtt{view}_H(r) = \alpha$, and $obs_H(t) = o_H$, and $\mathtt{view}_L(r) = \beta$ and $obs_L(t) = o_L$. Let s be the final state of r. Then we have $s \in K(\alpha, \pi, \beta)$. It is now immediate that $t \in \delta_{a_H,o_H,a_L,o_L}(K(\alpha, \pi, \beta))$. □

The following result shows that it suffices to consider strategies in which the choice of action depends only on the time and H's knowledge set, given the L view being excluded.

Lemma 3. *If there exists an H strategy π that excludes β, then there exists an H strategy π' that also excludes β, and has the property that for all H views α and α', if $K(\alpha, \pi', \beta) = K(\alpha', \pi', \beta)$ and $|\alpha| = |\alpha'|$ then $\pi'(\alpha) = \pi'(\alpha')$.*

Proof. Suppose that π excludes β. For purposes of the proof, note that we can assume without loss of generality that β is infinite — this helps to avoid mention of views longer than β as a separate case. (Note that it is equivalent to say that π excludes some prefix of β.)

Let f be any mapping from $\mathcal{V}_H$ to $\mathcal{V}_H$ such that for all $\alpha, \alpha' \in \mathcal{V}_H$ we have

1. $|f(\alpha)| = |\alpha|$,
2. $K(\alpha, \pi, \beta) = K(f(\alpha), \pi, \beta)$,
3. if $|\alpha| = |\alpha'|$ and $K(\alpha, \pi, \beta) = K(\alpha', \pi, \beta)$, then $f(\alpha) = f(\alpha')$.

Such a mapping always exists; intuitively, it merely picks, at each length, a representative $f(\alpha) \in [\alpha]_\sim$ of the equivalence classes of the equivalence relation defined by $\alpha \sim \alpha'$ if $|\alpha| = |\alpha'|$ and $K(\alpha, \pi, \beta) = K(\alpha', \pi, \beta)$.

Now define the mapping g on $\mathcal{V}_H$ as follows. Let $\alpha_0 = obs_H(s_0)$ be the only possible H view of length 0. For $\alpha \in \mathcal{V}_H$ of length 0, we define $g(\alpha) = \alpha$. For longer α, we define $g(\alpha a o) = f(g(\alpha))\pi(f(g(\alpha))o$. Also, define the strategy π' by $\pi'(\alpha) = \pi(f(g(\alpha)))$.

We claim that for all $\alpha \in \mathcal{V}_H$ we have $K(\alpha, \pi', \beta) = K(g(\alpha), \pi, \beta)$. The proof is by induction on the length of α. The base case is straightforward, since α_0 is consistent with all strategies, so $K(\alpha_0, \pi', \beta) = \{s_0\} = K(\alpha_0, \pi, \beta)$, and $K(\alpha, \pi', \beta) = \varnothing = K(\alpha, \pi, \beta)$ for $\alpha \neq \alpha_0$ with $|\alpha| = 0$. Suppose the claim holds for $\alpha \in \mathcal{V}_H$ of length i. Let $\alpha a o \in \mathcal{V}_H$. By induction and (2), $K(\alpha, \pi', \beta) = K(g(\alpha), \pi, \beta) = K(f(g(\alpha)), \pi, \beta)$. Let $\beta' a_L o_L$ be the prefix of β of length $|\alpha| + 1$. Since action $a = \pi'(\alpha) = \pi(f(g(\alpha))$, using Lemma 2, we have that

$$\begin{aligned}
K(\alpha a o, \pi', \beta) &= K(\alpha a o, \pi', \beta' a_L o_L) \\
&= \delta_{a,o,a_L,o_L}(K(\alpha, \pi', \beta')) \\
&= \delta_{a,o,a_L,o_L}(K(f(g(\alpha)), \pi, \beta')) \\
&= K(f(g(\alpha))ao, \pi, \beta' a_L o_L) \\
&= K(f(g(\alpha))ao, \pi, \beta) \\
&= K(g(\alpha a o), \pi, \beta).
\end{aligned}$$

To see that π' has the required property, if $K(\alpha, \pi', \beta) = K(\alpha', \pi', \beta)$ with $|\alpha| = |\alpha'|$, then we have $K(g(\alpha), \pi, \beta) = K(g(\alpha'), \pi, \beta)$. By (3) we have $f(g(\alpha)) = f(g(\alpha'))$. Therefore $\pi'(\alpha) = \pi(f(g(\alpha))) = \pi(f(g(\alpha'))) = \pi'(\alpha')$, by definition.

Since π excludes β, there exists a length n such that for all $\alpha \in \mathcal{V}_H$ with $|\alpha| = n$, we have $K(\alpha, \pi, \beta) = \varnothing$. Thus, we also have for all α of length n that $K(\alpha, \pi', \beta) = K(g(\alpha), \pi, \beta) = \varnothing$. This means that π' also excludes β. □

Based on Lemma 3, we construct a transition system $T(M) = (Q, q_0 \Rightarrow)$ that simultaneously searches for the strategy π and an L view β that is excluded by π. The components are defined by:

1. $Q = \mathcal{P}(S) \times \mathcal{P}(\mathcal{P}(S))$,
2. $q_0 = (\{s_0\}, \{\{s_0\}\})$,
3. the transition relation $\Rightarrow$ is defined by $(U, \mathcal{K}) \Rightarrow^{(\rho, a_L, o_L)} (U', \mathcal{K}')$ if
   (a) $\rho : \mathcal{K} \to A_H$, and $a_L \in A_L$ and $o_L \in O$,
   (b) $U' = \{t \mid$ there exists $s \in U$, and a transition $s \xrightarrow{(a'_H, a_L)} t$, with $a'_H \in A_H$ and $o_L = obs_L(t)\ \} \neq \varnothing$, and
   (c) $\mathcal{K}' = \{\delta_{a_H,o_H,a_L,o_L}(k) \mid k \in \mathcal{K}$ and $a_H = \rho(k)$ and $o_H \in O\}$.

Intuitively, the component U in a state $(U, \mathcal{K})$ is used to ensure that the view β that we construct is in fact possible in M. The component $\mathcal{K}$ represents a collection of all possible knowledge sets that H can be in at a certain point of time, while attempting to exclude β. More specifically, each set k in $\mathcal{K}$ corresponds to $\alpha \in \mathcal{V}_H$ such that $k = K(\alpha, \pi, \beta)$. In a transition, we both determine the next phase of π, by extending π so that $\pi(\alpha) = \rho(K(\alpha, \pi, \beta))$, and extend β to $\beta a_L o_L$. Moreover, an H strategy generated by $T(M)$ is only sensitive to H's knowledge set and lengths of runs, i.e., it satisfies that

$K(\alpha,\pi',\beta) = K(\alpha',\pi',\beta)$ and $|\alpha| = |\alpha'|$ implies $\pi'(\alpha) = \pi'(\alpha')$. In the above construct, ρ represents the local choice of H that depends only on the knowledge set of H.

The following result justifies the correspondence between the transition system $T(M)$ and NDS.

Lemma 4. *A machine M does not satisfy* NDS *iff $T(M)$ contains a path $q_0 \Rightarrow^* (U,\mathcal{K})$ to a state where $\mathcal{K} = \varnothing$.*

Proof. We first prove the implication from left to right. Suppose first that M does not satisfy NDS, witnessed by the fact that π excludes the possible L view $\beta = o_0 b_1 o_1 b_2, \ldots b_n o_n$. We may assume without loss of generality that no strict prefix of β is excluded. By Lemma 3, we may assume that π has the property that it takes the same value on H views α, α' that have the same length and have $K(\alpha,\pi,\beta) = K(\alpha',\pi,\beta)$. We construct a path

$$q_0 = (U_0,\mathcal{K}_0) \Rightarrow^{(\rho_1,b_1,o_1)} (U_1,\mathcal{K}_1) \Rightarrow^{(\rho_2,b_2,o_2)} \ldots \Rightarrow^{(\rho_n,b_n,o_n)} (U_n,\mathcal{K}_n)$$

in the transition system $T(M)$, by defining the functions $\rho_i : \mathcal{K}_{i-1} \to A_H$ for $i \geq 1$, and then deriving U_i from U_{i-1} using the equation in clause 3(b) of the definition of $T(M)$, and deriving $\mathcal{K}_i$ from $\mathcal{K}_{i-1}$ and ρ_i using the equation in clause 3(c). (This guarantees that each step satisfies all the conditions of the definition of $\Rightarrow$, except the requirement in 3(b) that $U' \neq \varnothing$; we check this below.) The construction will have the property that every $k \in \mathcal{K}_{i-1}$ is equal to some $K(\alpha,\pi,\beta)$ with $\alpha \in \mathcal{V}_H$ of length $i-1$. This means that we may define $\rho_i(k) = \pi(\alpha)$. Note that ρ_i is well-defined, by the assumption on π. More precisely, we claim that for each $i = 0 \ldots n$, $\mathcal{K}_i$ is a subset of the set $\{K(\alpha,\pi,o_0 b_1 o_1 \ldots b_i o_i) \mid |\alpha| = i\}$. Note that this means that if $k \in \mathcal{K}_n$ then $k = \varnothing$, for else we have an H view α of length $|\beta|$ such that $K(\alpha,\pi,\beta) \neq \varnothing$, which implies that π does not exclude β. Thus $\mathcal{K}_n = \varnothing$, as required for the right hand side of the result.

The proof of the claim is by induction on i. The base case of $n = 0$ is immediate from that fact that β is a possible view, so $K(obs_H(s_0),\pi,o_0) = \{s_0\}$. Suppose $k' \in \mathcal{K}_{i+1}$. We show $k' = K(\alpha',\pi,o_0 b_1 o_1 \ldots b_i o_i b_{i+1} o_{i+1})$ for some α' of length $i+1$. By definition of $\mathcal{K}_{i+1}$, there exists $k \in \mathcal{K}_i$, and $o_H \in O$, such that with $a_H = \rho(k)$, we have $k' = \delta_{b_{i+1},o_{i+1},a_H,o_H}(k)$. By induction, there exists $\alpha \in \mathcal{V}_H$ of length i such that $k = K(\alpha,\pi,o_0 b_1 o_1 \ldots b_i o_i)$. By Lemma 2, it follows that $k' = K(\alpha a_H o_H,\pi,o_0 b_1 o_1 \ldots b_i o_i b_{i+1} o_{i+1})$, as required.

It remains to show that $U_i \neq \varnothing$ for each $i = 1 \ldots n$. For this, note that since β is a possible L view, there exists a run $s_0 \xrightarrow{(a_1,b_1)} s_1 \xrightarrow{(a_2,b_2)} \ldots \xrightarrow{(a_n,b_n)} s_n$ such that $obs_L(s_i) = o_i$ for $i = 1 \ldots n$. A straightforward induction shows that for each i, we have $s_i \in U_i$, so in fact $U_i \neq \varnothing$, as required.

For the other direction, suppose that

$$q_0 = (U_0,\mathcal{K}_0) \Rightarrow^{(\rho_1,b_1,o_1)} (U_1,\mathcal{K}_1) \Rightarrow^{(\rho_2,b_2,o_2)} \ldots \Rightarrow^{(\rho_n,b_n,o_n)} (U_n,\mathcal{K}_n)$$

and $\mathcal{K}_n = \{\varnothing\}$. We construct a strategy π that excludes $\beta = obs_L(s_0) b_1 o_1 \ldots b_n o_n$. A straightforward induction using clause 3(b) of the definition of $T(M)$ shows that β is a possible L view in M. The construction of π is done inductively, by defining a sequence of strategies $\pi_0, \pi_1, \ldots, \pi_n$ such that if $i \leq j$ then π_i and π_j agree on all H views of length at most $i-1$. At each stage of the construction, we claim that for all H views α of length $i \geq 0$, if $K(\alpha,\pi_i,\beta) \neq \varnothing$ then $K(\alpha,\pi_i,\beta) \in \mathcal{K}_i$. Inductively, we let π_0 be any strategy and define $\pi_{i+1}(\alpha) = \rho_{i+1}(K(\alpha,\pi_i,\beta))$ if $|\alpha| = i$ and $K(\alpha,\pi_i,\beta) \neq \varnothing$, and $\pi_{i+1}(\alpha) = \pi_i(\alpha)$ otherwise. Evidently, π_{i+1} is well defined by the claim that $K(\alpha,\pi_i,\beta) \neq \varnothing$ then $K(\alpha,\pi_i,\beta) \in \mathcal{K}_i$. Also this definition plainly satisfies the condition that if $i \leq j$ then π_i and π_j agree on all views of length at most $i-1$. Note also that since $K_n = \{\varnothing\}$, by the claim there does not exist an H view α of length n such that $K(\alpha,\pi_n,\beta) \neq \varnothing$. It follows that π_n excludes β.

It therefore suffices to show that the definition satisfies the claim. Note that it holds trivially for any strategy if $i = 0$. Suppose that for all H views α of length i, if $K(\alpha,\pi_i,\beta) \neq \varnothing$ then $K(\alpha,\pi_i,\beta) \in \mathcal{K}_i$. Let $\alpha a_H o_H$ be an H view of length $i+1 \leq n-1$ with $K(\alpha a_H o_H,\pi_{i+1},\beta) \neq \varnothing$. Then also $K(\alpha,\pi_{i+1},\beta) \neq \varnothing$ and $\pi_{i+1}(\alpha) = a_H$. Since π_i and π_{i+1} agree on views of length at most $i-1$, we also have $K(\alpha,\pi_{i+1},\beta) = K(\alpha,\pi_i,\beta) \neq \varnothing$, so $K(\alpha,\pi_{i+1},\beta) \in \mathcal{K}_i$. By Lemma 2, we have that $K(\alpha a_H o_H,\pi_{i+1},\beta) = \delta_{b_{i+1},o_{i+1},a_H,o_H}(K(\alpha,\pi_{i+1},\beta)) \in \mathcal{K}_{i+1}$, as required. □

We obtain the claimed complexity bound from Lemma 4, simply by noting that it reduces NDS to a reachability problem in the transition system $T(M)$. Since the states of the system $T(M)$ can be represented in space $O(|S| \cdot 2^{|S|}) = 2^{O(|S|)}$, we obtain from Savitch's theorem that we can do the search in DSPACE($2^{O(|S|)}$).

4.2. EXPSPACE-hardness

To show that NDS is EXPSPACE-hard, we show how to encode the game BLIND–PEEK–PEEK of Reif [37]. We need only scheduled machines for the encoding, so the problem is EXPSPACE-hard already for this subclass.

4.2.1. The game BLIND–PEEK

BLIND–PEEK is a variant of the two-player game PEEK introduced by Stockmeyer and Chandra [39]. A PEEK game consists of a box with two open sides that contains horizontally stacked plates; the players sit at opposite sides of the box. Each plate has two positions, 'in' and 'out', and contains a knob at one side of the box, so that this plate can be controlled

by one of the players. At each step, one of the two players may grasp a knob from his side and push it 'in' or 'out'. The player may also pass. Both the top of the box and the plates have holes in various positions, and each hole is associated to a player. If, just after a move of player $a \in \{1, 2\}$, the plates are positioned so that for one of the player's holes in the top of the box, it is possible to peek through from the top of the box to the bottom (i.e., each plate has a hole positioned directly underneath the top hole), then player a wins. In PEEK, both players can observe the position of all plates at all times. BLIND–PEEK [37] (more formally, the game G^{2B} of that paper) is a modification of PEEK in which player 1's side of the box is partially covered, so that it is not possible for player 1 to see the positions of the plates controlled by player 2. We may represent the game formally as follows:

Definition 4.1. An instance G of the BLIND–PEEK game is given by a tuple $(n, n_1, \Phi_1, \Phi_2, \nu_0)$ where n and n_1 are natural numbers with $n_1 < n$,

$$\Phi_1 = \bigvee_{j=1}^{h_1} \gamma_j^1 \text{ and } \Phi_2 = \bigvee_{j=1}^{h_2} \gamma_j^2$$

are disjunctive normal form formulas over the set of atomic propositions $\{P_1, \ldots, P_n\}$, and $\nu_0 : [1..n] \to \{0, 1\}$ represents a boolean assignment to these propositions. The size of the instance is $O(n(h_1 + h_2))$.

Here h_i, for $i \in \{1, 2\}$, is the number of holes on the top of the box for each player. Intuitively, n gives the number of plates, and the propositions P_k for $1 \le k \le n$ correspond to the positions of the plates, which can be either *in* (P_k false) or *out* (P_k true), and a state of the game G is given by a mapping $\nu : [1..n] \to \{0, 1\}$, with P_k true at ν just when $\nu(k) = 1$. The total number of states in a game is thus exponential in the size of the game. The assignment ν_0 specifies the initial state of the game. The number n_1 specifies the number of plates associated to player 1; we take these to be plates $1..n_1$. The formula Φ_i gives the winning condition for player i. Each disjunct γ_j^i corresponds to one of the holes on the top of the box that is associated to player i. Which literals are in γ_j^i depends on how the hole in the top of the box aligns with a hole on the plates when these are in or out. If there is always an alignment with a hole on plate k then γ_j^i contains neither P_k nor $\neg P_k$. If there is an alignment only when the k-th plate is *out* then γ_j^i contains the literal P_k, and conversely, if there is an alignment only when the k-th plate is *in* then γ_j^i contains the literal $\neg P_k$. (If there is never an alignment then we may include both P_k and $\neg P_k$, but, obviously, we may just as well remove the hole from the game.)

Players 1 and 2 play in turn by moving one of their plates or passing. As the players' plate numbers partition the set $[1 \ldots n]$, we can denote the moves of the players $\texttt{move}_i$ with $1 \le i \le n$; if $i \in [1..n_1]$ it is a move of player 1, and a move of player 2 otherwise. We let $\texttt{Move}_1 = \{\texttt{move}_i \mid 1 \le i \le n_1\} \cup \{\texttt{Pass}\}$ and $\texttt{Move}_2 = \{\texttt{move}_i \mid n_1 + 1 \le i \le n\} \cup \{\texttt{Pass}\}$.

A *play* ϱ in G is an alternating sequence of player 1 and player 2 moves of the form

$$\varrho = \nu_0 \xrightarrow{\lambda_1} \nu_1 \xrightarrow{\lambda_2} \nu_2 \quad \cdots \quad \nu_{i-1} \xrightarrow{\lambda_i} \nu_i$$

where λ_j is a player 1 move in $\texttt{Move}_1$ when j is odd and a player 2 move in $\texttt{Move}_2$ when j is even. Moreover, if $\lambda_l = \texttt{Pass}$ then $\nu_l = \nu_{l-1}$ and if $\lambda_l = \texttt{move}_k$, then $\nu_l(k) = 1 - \nu_{l-1}(k)$ and $\nu_l(j) = \nu_{l-1}(j)$ for $j \ne k$.

A state ν is winning for player p if it satisfies the formula Φ_p. A play ϱ is *winning for player p* if it contains a state ν_k immediately after a move by player p that is winning for that player, and there is no earlier such winning state for the other player. Otherwise the play is undecided.

We are interested in the problem of deciding whether there is a winning strategy for player 1, i.e., a way for the player to choose their moves that guarantees, whatever the other player does, that player 1 will win. Strategies usually choose a next move based on what the player has been able to observe over a play of the game. In the case of the game G, the information directly visible to player 1 in a state is just the position of plates $1..n_1$. Player 2 sees the position of all plates. A player also remembers the sequence of moves they have played at their turn. At each step of the play, the player is also advised whether any player has won the game. (In a physical realization of the game, if a player were to peek through a hole they would be able to see which is the topmost plate that blocks it. We assume that the player does *not* get this information in the formal game. One can imagine a physical realization in which a referee peeks through the holes and announces the result.)

Since the winning condition is a discrete, state-based condition, deterministic strategies suffice. Moreover, note that, except for the information about who has won the game, the effect of the players' moves on the information directly visible to player 1 is deterministic: we can deduce the position of player 1's plates from the moves that player 1 has made so far in the game. Thus, every undecided play in which player 1 has made a particular sequence of moves yields the same view for player 1, and on a deterministic strategy, player 1 must make the same next move on all such plays. This means that we may represent a player 1 strategy simply by a (finite or infinite) sequence of moves $\Lambda = \lambda_1, \lambda_3, \lambda_5 \ldots$. Such *a strategy is winning for player 1* if every play with this sequence of moves by player 1 is winning for player 1.

The following result characterizes the complexity of BLIND–PEEK.

Theorem 5. *(See [37].) The game BLIND–PEEK is complete for EXPSPACE under log-space reductions.*

We use this result to show that NDS is EXPSPACE-hard. Given an instance G of BLIND–PEEK, we construct a synchronous system $M(G)$ of size polynomial in the size of G, with the following property: player 1 has a winning strategy in G iff there exists an L view v_L and an H strategy π that excludes v_L in $M(G)$.

Intuitively, the winning strategy Λ of player 1 in the game G will be encoded within the sequence of L actions contained in the view v_L. The role of the H strategy π in the machine $M(G)$ is to help in the verification that the strategy Λ is winning, by ensuring that the view v_L cannot occur when this is the case. As we cannot encode the exponential number of possible states of the BLIND–PEEK game G directly in the polynomial number of states of $M(G)$, we use an encoding trick, which is to represent the state of the game as the set of states of $M(G)$ that are consistent with the H view. Roughly, each such consistent state corresponds to one of the plates; there are some additional states for initialization and book-keeping related to the winning condition.

4.2.2. High level structure of $M(G)$

We let $n_2 = n - n_1$ be the number of plates that can be moved by player 2.

The machine $M(G)$ will be a scheduled machine (see Section 2.4), with deterministic schedule following the regular expression $\bot(LHL\bot H^{h_2})^\omega$. Here occurrences of H and L indicate which agent's action the transition is allowed to depend upon, and $\bot$ is for a system step that is independent of both agents H and L. We call each instance of the infinitely repeated block $LHL\bot H^{h_2}$ a *round*, and use indices, as in $L_1 H_0 L_2 \bot H_1 .. H_{h_2}$ to refer to the stages of the round.

The alphabet of actions for L and H are A_L and A_H defined by:

$$A_L^- = \{\texttt{move}_i \mid i \in [1 \dots n_1]\}$$
$$A_L = A_L^- \cup \{\texttt{checkwin}\}$$
$$A_H = \{\texttt{isOpen}_k \mid k \in [1..h_1]\} \cup \{\texttt{isBlocking}_k \mid k \in [1 \cdots n]\}.$$

Informally, the behaviour of $M(G)$ in each step is as follows:

1. In the first $\bot$ step of the schedule, the machine nondeterministically makes a transition to one of n subsystems, each of which monitors one particular plate of the game. Neither H nor L is able to see which subsystem they are actually in during subsequent transitions. The machine then moves into the cyclically repeated rounds.
2. The L_1 stage of each round allows L to perform one move of player 1 in the game G (using an action $\texttt{move}_i$) or to pass (using the action checkwin). This stage corresponds to a move according to a blindfold strategy of player 1.
3. In the following H_0 stage, agent H is given an opportunity to assert that the last L move has achieved a win of player 1, by specifying a hole j of player 1 and claiming that it is possible to peek through (using an action $\texttt{isOpen}_j$). H may also pass (using any other action).
4. The following L_2 stage allows L to check, by performing a "checkwin" action, if a winning state has been reached, as claimed by H sometime before. If it is so, then some of the L views will be ruled out. If it is still not a win, or H has not yet asserted a win, or H has made a mistake, this "checkwin" action will not rule out any L view. (Once a "checkwin" fails in this way, no L views will be excluded thereafter.)
5. The next stage $\bot$ simulates a move of player 2 in the game G.
6. During the last h_2 stages, agent H is given an opportunity to assert that the last player 2 move is not a win, and explain this claim by pointing, for each of the h_2 peek-holes of player 2, to a plate j that blocks that hole, using an action $\texttt{isBlocking}_j$.

The observations of the agents L, H in $M(G)$ are defined so that neither agent ever learns which plate is being simulated on the current run. Agent L observes the player 1 moves and a result of any "checkwin" actions. Agent H observes all moves by either player.

Intuitively, suppose player 1 has a winning strategy and agent L faithfully follows this strategy in the L_1 stage of each round. Suppose also that agent H, who knows every previous move of the play, always makes correct assertions about whether holes are open or blocked. Then L is guaranteed to be able to eventually make a successful "checkwin" and get some L views ruled out. To handle the case where H makes incorrect assertions, the construction ensures that no L view is eliminated if this happens. Thus, the statement that there is some H strategy that eliminates an L view corresponds to the statement that H has a way of making correct assertions in order to show that the player 1 strategy is winning.

4.2.3. States and observations of $M(G)$

Formally, the state space of $M(G)$ is defined as

$$\{s_0\} \cup (\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}) \cup (\mathbb{C} \times \mathbb{F}),$$

where s_0 is the initial state, and the components are as follows:

- $\mathbb{C} = \{L_1, H_0, L_2, \bot, H_1, \cdots, H_{h_2}\}$ encodes a clock that represents the current stage in a round of the cyclic part of the schedule,

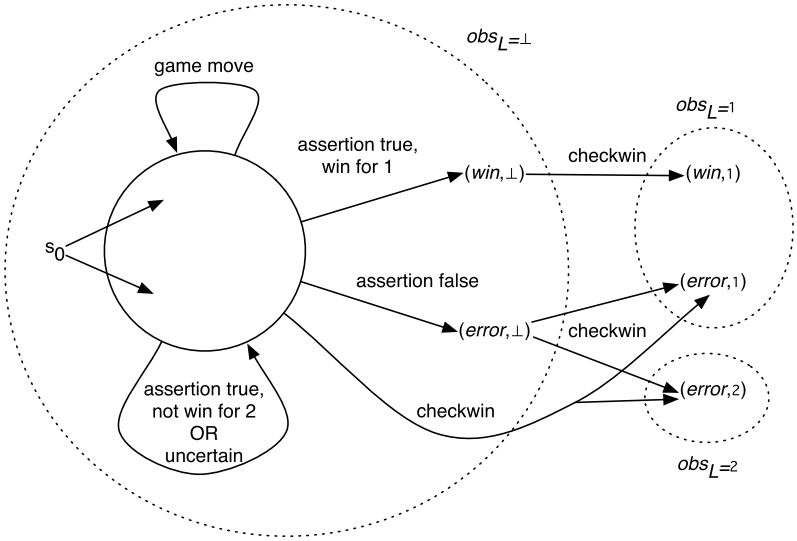

Fig. 5. High level structure of $M(G)$.

- $\mathbb{P} = \{1, \ldots n\}$ represents the plate being monitored,
- $\mathbb{B} = \{0, 1\}$ encodes whether the current plate is "in" or "out",
- $\mathbb{M} = \texttt{Move}_1 \cup \texttt{Move}_2 \cup \{\bot\}$ records the most recent move in the play (and $\bot$ for simulation steps that do not correspond to game steps), and
- $\mathbb{F} = \{(\mathit{win}, \bot), (\mathit{error}, \bot), (\mathit{win}, 1), (\mathit{error}, 1), (\mathit{error}, 2)\}$ records the result of claims made by H.

The observation mappings for H and L on these states are given by:

- $\mathit{obs}_L((c, r, 1)) = 1$ and $\mathit{obs}_L((c, r, 2)) = 2$, where $c \in \mathbb{C}$ and $r \in \{\mathit{win}, \mathit{error}\}$, and $\mathit{obs}_L(s) = \bot$ for all s not of this form.
- $\mathit{obs}(s_0) = \bot$, and $\mathit{obs}_H((c, i, k, a)) = a$ and $\mathit{obs}_H(s) = \texttt{end}$ for all $s \in \mathbb{C} \times \mathbb{F}$.

States of $M(G)$ of type $(c, i, k, a) \in \mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$ encode information about the effect of the play of the game so far on a particular plate: $c \in \mathbb{C}$ indicates the current stage of the simulation, $i \in [1..n]$ is a *monitored* plate, $k \in \mathbb{B}$ is the position of the plate i and $a \in \mathbb{M}$ is the most recent move made in the game, or $\bot$ if none.

States of $M(G)$ in $\mathbb{C} \times \mathbb{F}$ are used to capture the effect of assertions made by H relating to the winning conditions, and play a key role in ensuring that an L view is eliminated under the appropriate conditions. These states form a "terminal" part of the machine: it is not possible to return to the component $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$ from these states. The $\mathbb{C}$ component simply tracks the simulation stages. Fig. 5 sketches the way that the $\mathbb{F}$ components of these states are used to check winning conditions for the game and to generate observations.

Intuitively, at various stages of the simulation (viz., H_0 and $H_1 \ldots H_{h_2}$), agent H is allowed to make assertions about the state of the game. When H makes such an assertions, $M(G)$ checks whether they are true at the plate being simulated.

- If the assertion entails that there is not yet a win for player 2, and is true of the present plate, or does not concern the present plate, then we continue the simulation. (Specifically, this case occurs when the assertion is that a particular plate is blocking the hole for player 2 under consideration, and this is true or concerns another plate.)
- If the assertion entails a win for player 1 and is true of the present plate, then a transition is made to a state $(\mathit{win}, \bot)$. (Specifically, the assertion is that a particular player 1 hole is open, and this holds at the present plate.)
- The remaining possibility is that the assertion is false. In this case we make a transition to a state $(\mathit{error}, \bot)$. (We have this case when either the assertion is that a particular player 1 hole is open, but this is false at the present plate, or is that the present plate is blocking a particular player 2 hole, but this is false.)

When L eventually performs a `checkwin` action in the appropriate phase L_2, states with $(\mathit{error}, \bot)$ could produce either the observation 1 or 2. By contrast, states with $(\mathit{win}, \bot)$ produce only the observation 1. Thus, the *win* states result in a reduced set of views. Note that we can only check assertions locally at the current plate, and the winning condition for player 1 requires that a player 1 hole be unblocked at all plates. The way that the encoding handles this is via L's uncertainty about which plate is being monitored: if there is any plate at which the hole is blocked, then there will be a run consistent with L's observations, monitoring this plate, at which we have $(\mathit{error}, \bot)$. When L performs `checkwin` in this run it will obtain both observations 1 and 2, and the set of views is not reduced.

4.2.4. Transitions of $M(G)$

In general, transitions in a machine are labelled by joint actions (a_H, a_L) of H and L, but since $M(G)$ is a scheduled machine, at most one of these has any effect on the state. To simplify the presentation, we use the convention of writing $s \xrightarrow{b} t$ with $b \in A_u$, to indicate that the current transition is dependent only on u, and write $s \xrightarrow{\tau} t$ if the transition is independent of both H and L (i.e., a system step). To capture the scheduler, we define the function $next : \mathbb{C} \to \mathbb{C}$ so that it maps each element of the sequence $L_1, H_0, L_1, \bot, H_1 \dots H_{h_2}$ to the element next in the sequence, with $next(H_{h_2}) = L_1$.

To describe the transitions we use the following predicates relating to the winning conditions. When $i \in \{1, 2\}$ is a player, $j = 1..h_i$ is a hole associated to that player, $k \in \mathbb{P}$ is a plate and $b \in \mathbb{B}$ is a plate position, we define $\texttt{Open}^i_j(k, b)$, to be true just when either proposition P_k does not occur positively or negatively in γ^i_j, or P_k occurs positively in γ^i_j and $b = 1$ or $\neg P_k$ occurs in γ^i_j and $b = 0$. Intuitively, this says that when plate k is in position b, it is not blocking hole j.

We group the transitions according to the step of the schedule. We first describe transitions from states in $\{s_0\} \cup (\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M})$.

Initial Step $\bot$ The initial state of $M(G)$ is s_0. From this state, we non-deterministically choose one plate to be monitored. This transition does not depend on any agents and has the form

$$s_0 \xrightarrow{\tau} (L_1, i, \nu_0(i), \bot)$$

for $i \in [1 \dots n]$, where ν_0 is the initial state of G.

Stage L_1 At this stage, we simulate player 1's move of a plate j. The special action `checkwin` is used for the pass move. For each $\texttt{move}_j \in A_L$, and state (L_1, i, k, a) there is a transition of the form:

$$(L_1, i, k, a) \xrightarrow{\texttt{move}_j} (H_0, i, k', \texttt{move}_j)$$

with $k' = 1 - k$ if $j = i$ and $k' = k$ otherwise. We also add

$$(L_1, i, k, a) \xrightarrow{\texttt{checkwin}} (H_0, i, k, \texttt{Pass})$$

for the pass move.

Stage H_0 At this stage H may try to prove that player 1 can peek through some hole $j \in [1..h_1]$. To do this, it chooses an action $\texttt{isOpen}_j$. As it is a guess, H might be wrong. If H claims that player 1 can peek through hole j and this is right at the present plate, we reach a "winning" state in $\mathbb{C} \times \mathbb{F}$, otherwise an error state. From a plate simulation state (H_0, i, k, a) there is a transition

$$(H_0, i, k, a) \xrightarrow{\texttt{isOpen}_j} (L_2, r, \bot)$$

where $r = win$ if $\texttt{Open}^1_j(i, k)$, and $r = error$ otherwise. H may also intentionally choose not to declare a win, by performing any of its actions $\texttt{isBlocking}_j$ for $j \in \{1 \dots n\}$. This is captured by the transitions

$$(H_0, i, k, a) \xrightarrow{\texttt{isBlocking}_j} (L_2, i, k, \bot)$$

for $j \in \{1 \dots n\}$.

Stage L_2 At this stage, L can perform the action "`checkwin`" to check if H has proved a win by player 1. If the current state at this stage is a plate simulation state, then H has not yet claimed a win for player 1, and any past assertions made by H about player 2's winning condition were either true or irrelevant to the current plate. In this case, we do not have evidence for a player 1 win, so we do not wish to eliminate an L view. Thus, from states $s = (L_2, i, k, a)$, we have transitions

$$s \xrightarrow{\texttt{checkwin}} (\bot, error, 1) \quad \text{and} \quad s \xrightarrow{\texttt{checkwin}} (\bot, error, 2)$$

so that both observations 1 and 2 can be obtained. Agent L is also allowed to continue playing without checking for a win, by performing any of the actions $\texttt{move}_j$ with $j \in [1 \dots n_1]$. For this case we have a transition

$$(L_2, i, k, a) \xrightarrow{\texttt{move}_j} (\bot, i, k, \bot)\ .$$

(We discuss the case of transitions at this stage from states in $\mathbb{C} \times \mathbb{F}$ below.)

Stage $\perp$ At this stage, we simulate a move of a plate by player 2. The following transitions are in $M(G)$:

$$(\perp, i, k, a) \xrightarrow{\tau} (H_1, i, k', \texttt{move}_j)$$

for each $i \in \mathbb{P}$, $k \in \mathbb{B}$ and $j \in [(n_1+1)\dots n]$ (player 2's plates), and: $k' = 1-k$ if $i = j$ and $k' = k$ otherwise. To model a pass move by player 2 we also have a transition

$$(\perp, i, k, a) \xrightarrow{\tau} (H_1, i, k, \texttt{Pass}) .$$

Stages H_1 *to* H_{h_2} In these stages, H tries to prove that last move by player 2 was not a winning move for player 2. It does so by showing that all player 2 holes are blocked (by at least one plate). For each player 2 peek hole j, at stage H_j, agent H chooses a plate $i \in [1\dots n]$ and asserts that the hole is blocked by that plate using the action $\texttt{isBlocking}_i$. An incorrect assertion results in a transition to an error state $(c, \mathit{error}, \perp)$. In particular, if the current state is indeed a win of player 2, then some hole j is open at all plates, and any attempt H makes to assert that it closed at a plate causes a transition to an error state.

This is encoded by the following transitions. At state (H_j, i', k, a) with $j \in [1..h_2]$, we have

$$(H_j, i', k, a) \xrightarrow{\texttt{isBlocking}_i} (\mathit{next}(H_j), i', k, \perp)$$

when either $i \neq i'$ (plate i is not monitored in the current state), or $i = i'$ and not $\texttt{Open}^2_j(i,k)$ (the present plate is blocking player 2's hole j). On the other hand, if $i = i'$ and $\texttt{Open}^2_j(i,k)$, i.e., player 2's peek hole j is not blocked by plate i, we have the transition

$$(H_j, i', k, a) \xrightarrow{\texttt{isBlocking}_i} (\mathit{next}(H_j), \mathit{error}, \perp) .$$

As pointed out before, the construction is designed to ensure that H knows the exact state of the game and thus can always determine whether a peek hole is blocked or not. Since we are looking for a winning strategy for player 1, if there is a win by player 2 then the present L strategy has failed. We would therefore like to insist that H must play only $\texttt{isBlocking}_i$ actions at this stage of the process. To ensure this, we define transitions so that all H actions other than $\texttt{isBlocking}_i$ cause an error transition at these stages. That is, for all H actions $\texttt{isOpen}_j$ and states of the form (c, i, k, a) with $c \in \{H_1, \dots, H_{h_2}\}$, we have a transition

$$(c, i, k, a) \xrightarrow{\texttt{isOpen}_j} (\mathit{next}(c), \mathit{error}, \perp) .$$

This completes the description of transitions from states in $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$.

Transitions from $\mathbb{C} \times \mathbb{F}$ The behaviour of the machine on states in $\mathbb{C} \times \mathbb{F}$ was described informally above. The main effect of actions is from $\texttt{checkwin}$ actions performed at stage L_2. From states with observation $\perp$, the action $\texttt{checkwin}$ causes an L observation of 1 or 2. For the *win* states we have a transition

$$(L_2, \mathit{win}, \perp) \xrightarrow{\texttt{checkwin}} (\perp, \mathit{win}, 1)$$

and for the *error* states we have transitions

$$(L_2, \mathit{error}, \perp) \xrightarrow{\texttt{checkwin}} (\perp, \mathit{error}, 1) \quad \text{and} \quad (L_2, \mathit{error}, \perp) \xrightarrow{\texttt{checkwin}} (\perp, \mathit{error}, 2) .$$

For all other cases, i.e., for $c = L_2$ and an action $b = \texttt{move}_j$, or for $c \in \mathbb{C} \setminus \{L_2\}$ and an action b (appropriate to stage c), we have for all $r \in \{\mathit{win}, \mathit{error}\}$ a transition

$$(c, r, \perp) \xrightarrow{b} (\mathit{next}(c), r, \perp) .$$

States at which an observation of 1 or 2 has already been obtained by L act as sinks, except for scheduler moves, i.e., we have a transition

$$(c, r, x) \xrightarrow{b} (\mathit{next}(c), r, x)$$

for all $c \in \mathbb{C}$, $r \in \{\mathit{win}, \mathit{error}\}$ and $x \in \{1, 2\}$.

4.2.5. Correctness of the construction

We now give the argument for the correctness of the encoding.

We first characterize the views obtained by the agents in $M(G)$. In the case of agent L, the structure of the possible views follows straightforwardly from the fact that the transitions as defined above follow the structure indicated in Fig. 5. Until a `checkwin` action is performed by L at some stage L_2 state, L observes $\perp$. Once it performs that action at this stage, it will observe either 1 or 2 for the remainder of time. Thus, L views are prefixes of the sequences generated by the regular expressions

$$\perp A_L \perp (A_L \perp A_L \perp A_L^- \perp A_L \perp (A_L \perp)^{h_2})^* A_L \perp A_L \perp \texttt{checkwin}x\,(A_L x)^*$$

with $x = 1$ or $x = 2$. Here the expression $A_L \perp A_L \perp A_L^- \perp A_L \perp (A_L \perp)^{h_2}$ corresponds to a round in which L does not perform `checkwin` at stage L_2. If α is an L view, we write $\Lambda(\alpha)$ for the subsequence of player 1 actions performed at times when the simulation is at stage L_1, where we treat a `checkwin` at such a time as the action `Pass`.

The H views are prefixes of the sequences in the regular expression

$$\perp (A_H\,(\texttt{Move}_1 \cup \texttt{Move}_2 \cup \{\perp\}))^* (A_H\,\texttt{end})^*$$

where the observations obtained at stage H_0 are in $\texttt{Move}_1 \cup \{\perp\}$, the observations obtained at stage H_1 are in $\texttt{Move}_2 \cup \{\perp\}$, and all other observations before the first `end` are $\perp$.

We will show that there is a correspondence between plays of the game G and views β of H. In particular, given an H view β, let $\sigma(\beta) = \lambda_1 \ldots \lambda_n$ be the subsequence of elements of $\texttt{Move}_1 \cup \texttt{Move}_2$ appearing in observations. It follows from the definition of the transition relation that $\sigma(\beta)$ is an alternating sequence of player 1 and player 2 actions. Since the moves of G have a deterministic effect on the states of G, we obtain a play

$$\varrho(\beta) = \nu_0 \xrightarrow{\lambda_1} \nu_1 \xrightarrow{\lambda_2} \nu_2 \ldots \xrightarrow{\lambda_n} \nu_n$$

of the game G. We define $\nu(\beta)$ to be the final state ν_n of $\varrho(\beta)$.

Consider an action a of H performed at an H view β at stage c. This will be recorded in the view of H, which will have the form $\beta\,a\,o$ immediately after this action. We say that the action a is *truthful* at view β, if

- $\beta = \perp$ is the view obtained at the run s_0, or
- β contains an action $\texttt{isOpen}_j$ (intuitively, H has already discharged the obligation to prove that L wins), or
- the stage c is in $\{L_0, L_1, \perp\}$ (H makes no assertion at these stages), or
- $c = H_0$ and $a = \texttt{isOpen}_j$ and hole j of player 1 is open in state $\nu(\beta)$, i.e., $\nu(\beta)$ satisfies γ_j^1, or
- $c = H_0$ and $a = \texttt{isBlocking}_j$ for some j (this corresponds to no assertion by H), or
- $c = H_k$ for $k \in [1 \ldots h_2]$ and $a = \texttt{isBlocking}_i$ and not $\texttt{Open}_k^2(i, \nu(\beta)(i))$, i.e., player 2's hole k is blocked at plate i in state $\nu(\beta)$.

Note that we omit the case where $c = H_k$, $k \geq 1$ and $a = \texttt{isOpen}_j$; intuitively, this corresponds to an assertion of False by H, which has an obligation to prove that the play is winning for player 1, and is failing to do so in this instance. We say that the view β is truthful if for every prefix $\beta\,a\,o$, the action a is truthful at β.

We now show that, so long as the play $\varrho(\beta)$ is undecided, the knowledge set of H encodes the state $\nu(\beta)$. For agent u we write $K_u(\alpha)$ for the set of final states of runs r of $M(G)$ with $\texttt{view}_u(r) = \alpha$, representing agent u's knowledge of the state after obtaining view α. Given a stage c, a state ν of game G and $a \in \mathbb{M}$, we let $S(c, \nu, a)$ be the set defined by $S(c, \nu, a) = \{(c, i, \nu(i), a) \mid i \in [1..n]\}$. Note that ν can be recovered from the set $S(c, \nu, a)$.

Proposition 3. *Suppose that β is an H view in $M(G)$ of length at least 1 such that β does not contain the observation* `end` *and $\varrho(\beta)$ is an undecided play of game G. Let c be the stage reached at the end of β and let the final observation of β be a. Then if β is truthful, we have $K_H(\beta) = S(c, \nu(\beta), a)$.*

Proof. We proceed by induction on the length of β. If β has length 1, then it arises from a run

$$s_0 \xrightarrow{(a_H, a_L)} (L_1, i, \nu_0(i), \perp)$$

so we have $\beta = \perp a_H \perp$, and $\sigma(\beta)$ is the empty sequence, corresponding to the play $\varrho(\beta) = \nu_0$. (Since we are interested in views, we use the explicit form of runs here, with actions of both H and L given, rather than the shorthand form used above, which mentioned only actions of the scheduled agents and implies that action the other agents may take all possible values.) The runs consistent with β are

$$s_0 \xrightarrow{(a_H, a)} (L_1, j, \nu_0(j), \perp)$$

where $j \in \mathbb{P}$ and a is some action of L. It is immediate that the claim holds.

Inductively, assume that $\beta\, a_H\, o$ is a truthful H view in $M(G)$ such that $K_H(\beta) = S(c, \nu(\beta), a)$ for some $c \in \mathbb{C}$ and $a \in \mathbb{M}$. We need to show that $K_H(\beta\, a_H\, o) = S(c', \nu(\beta\, a_H\, o), a')$ for some $c' \in \mathbb{C}$ and $a' \in \mathbb{M}$. Note that

$$K_H(\beta\, a_H\, o) = \{t \mid s \in K_H(\beta) \text{ and } a \in A_L \text{ and } s \xrightarrow{(a_H,a)} t \text{ and } obs_H(t) = o\}\ .$$

We consider each of the possible cases of the stage c.

Stage $c = L_1$. Here we have either $o = \texttt{move}_j$ or $o = \texttt{Pass}$, which records the action of L, and the transition does not depend on a_H. Note that $\sigma(\beta\, a_H\, o) = \sigma(\beta)\, o$ in this case. Let $\nu(\beta) \xrightarrow{o} \nu'$, so that $\nu(\beta\, a\, o) = \nu'$. For each plate $i \in \mathbb{P}$, we have $(L_1, i, \nu(i), a) \in K_H(\beta)$, and there is a unique transition

$$(L_1, i, \nu(i), a) \xrightarrow{(a_H,o)} (H_0, i, \nu'(i), o)$$

yielding observation o at the next state. It follows that $K_H(\beta\, a_H\, o) = S(H_0, \nu(\beta\, a_H\, o), o)$.

Stage $c = H_0$. Here we have $o = \bot$, and $\sigma(\beta\, a_H\, o) = \sigma(\beta)$ and $\nu(\beta\, a\, o) = \nu(\beta)$. Since $\varrho(\beta\, a\, o)$ is undecided, $\nu(\beta)$ is not a winning position for player 1. Because $\beta\, a_H\, o$ is truthful, we cannot have that a_H is $\texttt{isOpen}_j$ for any j, so we must have that $a_H = \texttt{isBlocking}_j$ for some j. For each plate $i \in \mathbb{P}$, we have $(H_0, i, \nu(\beta)(i), a) \in K_H(\beta)$, and there is a transition

$$(H_0, i, \nu(\beta)(i), a) \xrightarrow{(a_H,a')} (L_1, i, \nu(\beta)(i), \bot)$$

yielding observation $\bot$ at the next state for each action a' of L. It follows that $K_H(\beta\, a_H\, o) = S(H_0, \nu(\beta\, a_H\, o), o)$.

Stage $c = L_2$. Since β does not contain $\texttt{end}$, here we have $o = \bot$, and L cannot have performed the action $\texttt{checkwin}$. Also $\sigma(\beta\, a_H\, o) = \sigma(\beta)$ and $\nu(\beta\, a\, o) = \nu(\beta)$. Thus, from each $(L_2, i, \nu(\beta)(i), a) \in K_H(\beta)$, we have a transition

$$(L_2, i, \nu(\beta)(i), a) \xrightarrow{(a_H,a')} (\bot, i, \nu(\beta)(i), \bot)$$

from which we obtain that $K_H(\beta\, a_H\, o) = S(\bot, \nu(\beta\, a_H\, o), \bot)$.

Stage $c = \bot$. Here we have that $o = \texttt{Pass}$ or $o = \texttt{move}_i$ for some plate $i = n_1 + 1 \ldots n$ of player 2. Thus, $\sigma(\beta\, a_H\, o) = \sigma(\beta)\, o$ and if $\nu(\beta) \xrightarrow{o} \nu'$ then $\nu(\beta\, a\, o) = \nu'$. For each plate $i \in \mathbb{P}$, we have $(\bot, i, \nu(\beta)(i), a) \in K_H(\beta)$, and there is a unique transition

$$(\bot, i, \nu(\beta)(i), a) \xrightarrow{(a_H,o)} (H_1, i, \nu'(i), o)$$

yielding observation o. It follows that $K_H(\beta\, a_H\, o) = S(H_1, \nu(\beta\, a_H\, o), o)$.

Stage $c = H_i$ *for* $i = 1 \ldots h_2$. Here we have $o = \bot$, and $\sigma(\beta\, a_H\, o) = \sigma(\beta)$ and $\nu(\beta\, a\, o) = \nu(\beta)$. Since $\beta\, a_H\, o$ is truthful and $\sigma(\beta\, a_H\, o)$ is undecided, the position is not winning for player 2. Thus, we must have that $a_H = \texttt{isBlocking}_j$ for some j such that not $\texttt{Open}_i^2(j, \nu(\beta)(j))$. For each plate $i' \in \mathbb{P}$, we have $(H_i, i', \nu(\beta)(i'), a) \in K_H(\beta)$, and there is a transition

$$(H_i, i', \nu(\beta)(i), a) \xrightarrow{(a_H,a)} (\mathit{next}(H_i), i', \nu(\beta)(i'), \bot)$$

yielding observation $\bot$ for every L action a. It follows that $K_H(\beta\, a_H\, o) = S(H_1, \nu(\beta\, a_H\, o), o)$. □

In fact, we can show this characterization of $K_H(\beta)$ in one further case, corresponding to the state of the simulation just after player 1 plays a winning move.

Proposition 4. *Suppose that β is an H view in $M(G)$ of length at least 1 such that β does not contain the observation* $\texttt{end}$ *and $\varrho(\beta)$ is a play in which the last move is a move of player 1 by which player 1 wins the game. Assume that no shorter prefix of β has this property and let the final observation of β be a. Then if β is truthful, we have $K_H(\beta) = S(H_0, \nu(\beta), a)$.*

Proof. The minimality constraint on β implies that $\beta = \beta' a_H o$ where β' is at stage L_1, that $\varrho(\beta')$ is not a winning play for either player, and that $o \in \texttt{Move}_1$ is a move of player 1 such that $\varrho(\beta' a_H o) = \varrho(\beta') \xrightarrow{o} \nu(\beta)$. By Proposition 3, we obtain that $K_H(\beta') = S(L_1, \nu(\beta), a)$ for some a. The argument for the case of $c = L_1$ in the proof of Proposition 4 now yields the conclusion. □

Proposition 5. *Suppose that r is a run of $M(G)$ and $s \in K_H(\texttt{view}_H(r))$ is a state in $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$. Then there exists a run r' of $M(G)$ with final state s such that $\texttt{view}_H(r') = \texttt{view}_H(r)$ and $\texttt{view}_L(r') = \texttt{view}_L(r)$.*

Proof. By induction on the length of r. For $r = s_0$, the statement is trivial, since we must have $K_H(\mathtt{view}_H(r)) = \{s_0\}$. For another base case, suppose r is the initial step of a run. In this case,

$$r = s_0 \xrightarrow{(a_H, a_L)} (L_1, i, \nu_0(i), \bot)$$

and $\mathtt{view}_H(r) = \bot a_H \bot$ and $\mathtt{view}_L(r) = \bot a_L \bot$. If $s \in K_H(\mathtt{view}_H(r))$, then we must have $s = (L_1, j, \nu_0(j), \bot)$ for some j. We may take

$$r' = s_0 \xrightarrow{(a_H, a_L)} (L_1, j, \nu_0(j), \bot)$$

and this has the required properties.

For the induction, let

$$r = r_1 \xrightarrow{(a_H, a_L)} t$$

where the result holds for r_1, which is at stage c. Let $s \in K_H(\mathtt{view}_H(r))$. Since $s \in \mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$, the final observation of $\mathtt{view}_H(r)$ is not $\mathtt{end}$, and it follows that $t \in \mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$ also. We have that $s = (next(c), i, k, a)$ for some i, k, a, and arises in $K_H(\mathtt{view}_H(r))$ from some run

$$r_2 \xrightarrow{(a'_H, a'_L)} s$$

with $\mathtt{view}_H(r_2) = \mathtt{view}_H(r_1)$ and $a'_H = a_H$ and $obs_H(s) = obs_H(t)$. Necessarily, the final state of r_2 is in $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$. Moreover, it is in $K_H(\mathtt{view}_H(r_2)) = K_H(\mathtt{view}_H(r_1))$. By the induction hypothesis, there exists a run r_3 ending in the same final state as r_2, with $\mathtt{view}_H(r_3) = \mathtt{view}_H(r_1)$ and $\mathtt{view}_L(r_3) = \mathtt{view}_L(r_1)$. We consider the possibilities for the scheduler step c of r_1:

- *Case* $c = L_1$: Note that at this stage, the final L action in r can be deduced from $obs_H(s) = obs_H(t)$, so in fact we have $a'_L = a_L$ also. Let

  $$r' = r_3 \xrightarrow{(a_H, a_L)} s\,.$$

  This is a run because r_3 and r_2 have the same final state, and the final transition in r' is identical to the final transition of the run r_2. Then $\mathtt{view}_H(r') = \mathtt{view}_H(r_3)\, a_H\, obs_H(s) = \mathtt{view}_H(r_1)\, a_H\, obs_H(s) = \mathtt{view}_H(r)$ and $\mathtt{view}_L(r') = \mathtt{view}_L(r_3)\, a_L \bot = \mathtt{view}_L(r_1)\, a_L \bot = \mathtt{view}_L(r)$, as required.
- *Case* $c = H_k$ *for* $k = 0..h_2$: Transitions at these stages are independent of L, so we can switch the action of L in any transition label while keeping the states the same. So

  $$r' = r_3 \xrightarrow{(a_H, a_L)} s$$

  is a run and satisfies the required properties.
- *Case* $c = L_2$: Here it follows from $t \in \mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$ that $a_L \neq \mathtt{checkwin}$. Hence $a_L = \mathtt{move}_j$ for some j, and $obs_H(t) = obs_H(s) = \bot$. For the same reasons, $a'_L = \mathtt{move}_{j'}$ for some j'. The transitions for $\mathtt{move}_j$ and $\mathtt{move}_{j'}$ at this stage are identical. We may therefore take

  $$r' = r_3 \xrightarrow{(a_H, a_L)} s$$

  and this is a run and satisfies the required properties.
- *Case* $c = \bot$: Here transitions are independent of both players, so

  $$r' = r_3 \xrightarrow{(a_H, a_L)} s$$

  is a run and satisfies the required properties.

This completes the proof of the inductive case. □

We can now prove the key result that shows that G has a winning strategy for player 1 iff $M(G)$ satisfies $\mathtt{NDS}$.

Lemma 5. *There exists a winning strategy for player 1 in G iff there is an L view α of $M(G)$ and an H strategy π that excludes α in $M(G)$.*

Proof. *Only If Part.* Assume player 1 has a winning strategy in G. As argued before, this strategy can be given by the list $\Lambda = \lambda_1, \lambda_3 \ldots$ of moves of player 1. The number of moves player 1 needs to win is bounded: indeed, in every play of the game, a winning position for player 1 is eventually reached, and no winning position for player 2 is reached before this

position. By Koenig's lemma, there exists a number N such in all plays of the game compatible with Λ, player 1 has won the game at the latest, just after the N-th move. Thus, we may assume that $\Lambda = \lambda_1, \lambda_3 \dots \lambda_N$.

We can prove that there exists an H strategy π in $M(G)$ that excludes the L view

$$\alpha = \bot\, a_0 \bot a_1 \bot (a_0 \bot)^{3+h_2}\, a_3 \bot (a_0 \bot)^{3+h_2} \dots a_N \bot a_0 \bot \texttt{checkwin}\, 2,$$

where a_0 denotes any letter in A_L^-, and each a_i for i odd is the L action that corresponds to λ_i at stage L_1, i.e., $a_i = \lambda_i$ if $\lambda_i = \texttt{move}_j$ for some j, and $a_i = \texttt{checkwin}$ if $\lambda_i = \texttt{Pass}$. The strategy H is defined as follows. For H views β such that the sequence of player 1 moves in $\sigma(\beta)$ is a prefix of Λ, we let $\pi(\beta)$ be any truthful action of H at β. In all other cases, $\pi(\beta)$ is chosen arbitrarily.

We first need to show that π is well-defined. For this, we need to show that H is able to act truthfully whenever the sequence of player 1 moves in $\sigma(\beta)$ is a prefix of Λ. Suppose, therefore, that the sequence of player 1 moves in $\sigma(\beta)$ is a prefix of Λ. Then the play $\varrho(\beta)$ is not winning for player 2, since Λ is a winning strategy for player 1. There are two possibilities: the play is undecided, or the play is winning for player 1. If the play $\varrho(\beta)$ is undecided, then by Proposition 3, we have that $K_H(\beta) = S(c, \nu(\beta), a)$. Since $\nu(\beta)$ is, by definition, the final state of G reached in the play $\sigma(\beta)$, if $\sigma(\beta)$ ends in a move of player 1 then $\nu(\beta)$ is not a winning state for player 1, and if $\sigma(\beta)$ ends in a move of player 2 then $\nu(\beta)$ is not a winning state for player 2. In either case, depending on the stage, it is possible to select an action that is truthful at β.

In the other case, the play $\varrho(\beta)$ is already winning for player 1. Let β' be the smallest prefix of β such that $\sigma(\beta')$ is winning for player 1. Since the last action in $\sigma(\beta')$ is a move of player 1 (we can assume without loss of generality that the game is undecided at the initial state), must have that β' is at stage H_0, and $\nu(\beta)$ is a winning state for player 1. In the case that $\beta' = \beta$, we choose $\pi(\beta)$ to be any action $\texttt{isOpen}_j$ such that $\nu(\beta)$ satisfies γ_j^1, i.e., player 1's hole j is open at all plates in $\nu(\beta)$. This is then a truthful action at β. In all other cases, we choose $\pi(\beta)$ arbitrarily. (Note that, by definition, after H's first $\texttt{isOpen}_j$, any choice of H action is truthful.)

We now argue that π excludes view α. To the contrary, suppose that r is run consistent with π and $\texttt{view}_L(r) = \alpha$. Consider $\beta = \texttt{view}_H(r)$, and write this as $\beta = \beta_1\, a_H\, o$. Then the sequence of player 1 moves in $\sigma(\beta_1)$ is Λ. Since Λ is a winning strategy for player 1, the play $\varrho(\beta_1)$ is a winning play for player 1. Consider the shortest prefix β_2 of β_1 such that $\sigma(\beta_2)$ is a winning play for player 1. Then β_2 is at stage H_1, and there is at least one H action a and observation o' such that $\beta_2\, a\, o'$ is a prefix of β. (In the worst case, $\beta_2 = \beta_1$ and $a = a_H$.) By construction of π, a is an action $\texttt{isOpen}_j$ that is truthful at β_2. Using Proposition 4, $K_H(\beta_2) = S(H_1, \nu(\beta_2), a')$ for some a'. Because $\texttt{isOpen}_j$ is truthful at β_2, we obtain that $K_H(\beta_2\, a\, o') = \{(\bot, \mathit{win}, \bot)\}$.

In particular, the prefix r_2 of r with $\texttt{view}_H(r_2) = \beta_2\, a\, o'$ has final state $(\bot, \mathit{win}, \bot)$. Since L does not perform $\texttt{checkwin}$ at stage L_2 in the interim, the prefix r_1 of r with $\alpha = \texttt{view}_L(r_1)\,\texttt{checkwin}\,2$ has final state $(L_2, \mathit{win}, \bot)$. But then we get that the final state of r, after L performs $\texttt{checkwin}$, is $(\bot, \mathit{win}, 1)$, which yields an L observation of 1 rather than the final observation 2 of α. This is a contradiction.

If Part. Assume there is no winning strategy for player 1 in G. We show that there is no H strategy that can exclude any L view. To the contrary, assume an L view α that is excluded by an H strategy π. The following must hold:

1. The view α contains a $\texttt{checkwin}$ action at stage L_2. Indeed, if no $\texttt{checkwin}$ action at stage L_2 occurs, then all L observations in the view must be $\bot$. For any such sequence of L actions, there is always a run consistent with π yielding L observation $\bot$ throughout, so that, contrary to assumption, α is not excluded by π.
2. The view α is not of the form $\alpha_1\,\texttt{checkwin}\,1(A_L\,1)^*$, with α_1 being a view at stage L_2, and containing no prior $\texttt{checkwin}$ action at stage L_2. There is always a run consistent with π that yields such a view. There are two possibilities. If the final state s of a run yielding L view α_1 is in $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$, then the $\texttt{checkwin}$ action extends this run to one yielding L view $\alpha_1\,\texttt{checkwin}\,1$ by means of the stage L_2 transition

   $$s \xrightarrow{\texttt{checkwin}} (\bot, \mathit{win}, 1)\ .$$

   Otherwise, the state s is in $\mathbb{C} \times \mathbb{F}$, and must be of the form $(L_2, r, \bot)$, for $r \in \{\mathit{win}, \mathit{error}\}$, since there has not yet been a $\texttt{checkwin}$ at stage L_2. In this case, we obtain observation 1 at the next step by means of a transition

   $$(L_2, r, \bot) \xrightarrow{\texttt{checkwin}} (\bot, r, 1)$$

   for both possible values of r. The resulting states with observation 1 are sinks, so we can extend these runs to obtain a run with L view $\alpha_1\,\texttt{checkwin}\,1(A_L\,1)^*$. (In either case, we may take the H action in the final transition to be the action prescribed by π, since this transition is independent of H.)

It follows that view α is in the regular set $\alpha_1\,\texttt{checkwin}\,2(A_L\,2)^*$, with α_1 being a view at stage L_2, and containing no prior $\texttt{checkwin}$ action at stage L_2.

Let $\Lambda = \lambda_1 \lambda_3 \cdots \lambda_N$ be the player 1 moves of G corresponding to the actions taken by L in α at each L_1 stage. (These are the same as the L actions, except that we treat $\texttt{checkwin}$ at stage L_1 as corresponding to $\texttt{Pass}$.) The sequence Λ is a player 1 strategy in G. This strategy cannot be winning for player 1 as we have assumed that this player has no winning strategy in G. Thus, there exists some sequence $\lambda_2 \dots \lambda_{N-1}$ of player 2 moves such that the play

$$\varrho = \nu_0 \xrightarrow{\lambda_1} \nu_1 \xrightarrow{\lambda_2} \nu_2 \xrightarrow{\lambda_3} \dots \xrightarrow{\lambda_N} \nu_N$$

is not winning for player 1. Let r_1 be a run consistent with π such that $\mathtt{view}_L(r_1) = \alpha_1$ and at the m-th occurrence of stage $\perp$, we take the transition

$$(\perp, i, k, a) \xrightarrow{\tau} (H_1, i, k', \lambda_{2m})$$

corresponding to move λ_{2m} by player 2. (The choices of L actions in this run come from α_1, and the choices of H actions are fixed by the strategy π. The only nondeterminism remaining is in the initial step, where we choose the plate i to be monitored in the simulation. Since we will work at the level of the H view, any choice suffices.) Let $\beta_1 = \mathtt{view}_H(r_1)$ be the H view obtained along this run. Note that by construction of r_1, we obtain that $\varrho(\beta_1) = \varrho$ is the play which is not winning for player 1.

We now argue that the view β_1 is truthful and the play $\varrho(\beta_1)$ is also not winning for player 2. More precisely, we claim that for every prefix $\beta' a_H o$ of β_1, we have (1) the action $a_H = \pi(\beta')$ is truthful at β' and (2) the play $\varrho(\beta' a_H o)$ is not winning for player 2. We proceed by induction, assuming that β' is truthful and $\varrho(\beta')$ is not winning for player 2. Note that since $\varrho(\beta')$ is also not winning for player 1, we obtain by Proposition 3 that $K_H(\beta') = S(c, \nu(\beta'), a)$ for some $c \in \mathbb{C}$ and $a \in \mathbb{M}$. We consider the possible cases for the stage c:

1. If $c = L_1$, then $\varrho(\beta' a_H o) = \varrho(\beta') \xrightarrow{\lambda_m} \nu_m$ for some player 1 move λ_m. Since $\varrho(\beta')$ is not winning for player 2, an extension by a player 1 move also cannot be winning for player 2. But also a_H is trivially truthful at β', so both (1) and (2) hold.
2. If $c = H_0$, then $\varrho(\beta' a_H o) = \varrho(\beta')$ is not winning for player 2 by assumption, so we have (2). For (1), note that if $a_H = \mathtt{isBlocking}_j$ for some j then a_H is trivially truthful at β'. We show that the other case, where $a_H = \mathtt{isOpen}_j$ for some j, is not possible, because it leads to a contradiction. Note $\varrho(\beta')$ is also not a winning play for player 1, so $\nu(\beta')$ is not a winning position for player 1, and there exists a plate $i \in \mathbb{P}$ for which not $\mathtt{Open}^1_j(i, \nu(\beta')(i))$. This means that for the state $s = (c, i, \nu(\beta')(i), a) \in S(c, \nu(\beta'), a) = K_H(\beta')$, we have a transition

   $$s \xrightarrow{\mathtt{isOpen}_j} (L_1, \mathit{error}, \perp) .$$

   It follows using Proposition 5 that there exists a run r' ending in state $(L_1, \mathit{error}, \perp)$ with $\mathtt{view}_H(r') = \beta' a_H o$ and $\mathtt{view}_L(r')$ a prefix of α_1. The run r' is necessarily consistent with π because β is consistent with π. Following the actions dictated for L and H by α_1 and π, respectively, we may extend this to a longer run r'_1, still consistent with π, with $\mathtt{view}_L(r'_1) = \alpha_1$, also ending in state $(L_1, \mathit{error}, \perp)$. But then the next $\mathtt{checkwin}$ step allows a transition to $(\perp, \mathit{error}, 2)$, and we obtain a run with L view α, a contradiction.
3. If $c = L_2$, then a_H is trivially truthful, and $\varrho(\beta' a_H o) = \varrho(\beta')$ is not winning for player 2 by assumption.
4. For stages $c = H_k$ with $k \in \{1 \dots h_2\}$, we have that $\varrho(\beta' a_H o) = \varrho(\beta')$. It is immediate that $\varrho(\beta' a_H o)$ is not winning for either player, and it remains to show that a_H is truthful at β'.
   As noted above, the assumption that β' is truthful, together with the assumption that $\varrho(\beta')$ is not a winning play for either player, implies that $K_H(\beta') = S(c, \nu(\beta'), a)$, by Proposition 3. We will show that the desired conclusion that a_H is truthful at β' follows from the weaker assumption that β' is truthful and $K_H(\beta') = S(c, \nu(\beta'), a)$: this helps with the argument for case $c = \perp$, which is handled below.
   Note first that a_H cannot be $\mathtt{isOpen}_j$, since the final state of the prefix r' of r with $\mathtt{view}_H(r') = \beta'$ is in $S(c, \nu(\beta'), a)$, hence in $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$, so the action $\mathtt{isOpen}_j$ results in a transition to the state $(\mathit{next}(c), \mathit{error}, \perp)$ in r. It follows that the final state of r_1 is $(L_2, \mathit{error}, \perp)$, and then the subsequent action $\mathtt{checkwin}$ produces a run consistent with π with view α, contrary to the assumption that π excludes α. Hence $a_H = \mathtt{isBlocking}_j$ for some j.
   Suppose that $\mathtt{Open}^2_k(j, \nu(\beta')(j))$. Since $K_H(\beta') = S(c, \nu(\beta'), a)$, we have that $(c, j, \nu(\beta')(j), a) \in K_H(\beta')$. By Proposition 5, there exists a run r' ending in state $(c, j, \nu(\beta')(j), a)$ with $\mathtt{view}_H(r') = \beta'$ and $\mathtt{view}_L(r')$ a prefix of α_1. The transition

   $$(c, j, \nu(\beta')(j), a) \xrightarrow{\mathtt{isBlocking}_j} (\mathit{next}(c), \mathit{error}, \perp)$$

   extends this to a run whose L view remains a prefix of α_1, and by following strategy π and the remaining L actions in α_1 we may continue to extend to the point where we obtain a run r'_1 with $\mathtt{view}_L(r'_1) = \alpha_1$ and final state $(L_2, \mathit{error}, \perp)$. But then the next $\mathtt{checkwin}$ step allows a transition to $(\perp, \mathit{error}, 2)$, and we obtain a run consistent with π with L view α, a contradiction. Thus, in fact, we must have not $\mathtt{Open}^2_k(j, \nu(\beta')(j))$, so that a_H is truthful at β', as required.
   For the purposes of the next case, we make one further conclusion. Note that by definition of the transitions for $\mathtt{isBlocking}_j$ at stage H_k, we get from $K_H(\beta') = S(c, \nu(\beta'), a)$ that $K_H(\beta' a_H o) = S(\mathit{next}(c), \nu(\beta'), o) = S(\mathit{next}(c), \nu(\beta' a_H o), o)$, so we preserve the weakened assumption. Thus, since the above argument applies for all $k = 1 \dots h_2$, we have that for all such k, there exists j such that not $\mathtt{Open}^2_k(j, \nu(\beta')(j))$. That is, no hole of player 2 is open in $\nu(\beta')$. It follows that $\nu(\beta')$ cannot be a winning position of player 2.
5. If $c = \perp$, then a_H is trivially truthful, and $\varrho(\beta' a_H o) = \varrho(\beta') \xrightarrow{\lambda_m} \nu_m$, where $o = \lambda_m$ is a move of player 2. As noted above, $K_H(\beta') = S(c, \nu(\beta'), a)$ for some a. The transitions for case $c = \perp$ then imply that $K_H(\beta' a_H o) = S(c, \nu(\beta' a_H o), o)$. We

therefore satisfy the weakened assumption for the stages $H_1 \dots H_{h_2}$ in the previous case. It therefore follows using the argument of the previous case that $\nu(\beta' a_H o) = \nu_m$ cannot be a winning position of player 2. Thus, from the assumption that $\varrho(\beta')$ is not winning for either player, we obtain that $\varrho(\beta' a_H o)$ is not winning for either player.

This completes the argument that β_1 is truthful and $\varrho(\beta_1)$ is not a winning play for either player. By Proposition 3, we obtain that $K_H(\beta_1) = S(c, \nu(\beta_1), a)$ for some a. In particular, the final state of r_1 must be in $\mathbb{C} \times \mathbb{P} \times \mathbb{B} \times \mathbb{M}$, and the next `checkwin` action then results in a run consistent with π with L view α, a contradiction. □

Again, we point out that the hardness result holds for scheduled machines already.

5. Synchronous bisimulation-based notions

In this section we establish the result:

Theorem 6. *For the class of finite state synchronous machines,* `RES` *is in PTIME.*

The following Lemma shows that in searching for an unwinding relation on a machine M, it suffices to consider equivalence relations on the reachable states of M.

Lemma 6.

1. *If there exists a synchronous unwinding relation on M, then there exists a largest such relation, which is transitive.*
2. *If all states in M are reachable then the largest synchronous unwinding relation (if one exists) is an equivalence relation.*
3. *A system satisfies* `RES` *iff its restriction to its reachable states satisfies* `RES`.

Proof.

1. First we show that the set of synchronous unwinding relations on M is closed under union. Let $\sim_1$ and $\sim_2$ be two unwinding relations on M. Clearly Items 1 and 2 of Definition 2.5 hold for $\sim_1 \cup \sim_2$. Item 3 holds as well as if $s \sim_1 \cup \sim_2 t$ then either $s \sim_1 t$ or $s \sim_2 t$ holds. Assume $s \sim_1 t$, then as $\sim_1$ is an unwinding relation, by Item 3 of Definition 2.5, it follows that for all $a_1, a_2 \in A_H$ and $a_3 \in A_L$, if $s \xrightarrow{(a_1,a_3)} s'$ there is some t' such that $t \xrightarrow{(a_2,a_3)} t'$ and $s' \sim_1 t'$, which implies $s' \sim_1 \cup \sim_2 t'$. This implies that there exists a largest unwinding relation.
   Second, the composition of two unwinding relations is an unwinding relation. Again Items 1 and 2 of Definition 2.5 hold for $\sim_1 \circ \sim_2$. Assume $s \sim_1 \circ \sim_2 t$. In this case there is some x such that $s \sim_1 x$ and $x \sim_2 t$. As $\sim_1$ is an unwinding relation, by Item 3 of Definition 2.5, for any $a_1, a_2 \in A_H$ and $a_3 \in A_L$, if $s \xrightarrow{(a_1,a_3)} s'$, there is some x' such that $x \xrightarrow{(a_2,a_3)} x'$ and $x' \sim_1 t'$. As $x \sim_2 t$, and as $\sim_2$ is an unwinding relation, Item 3 Definition 2.5 applied with $a_1 = a_2$ implies there exists some t' such that $t \xrightarrow{(a_2,a_3)} t'$ and $x' \sim_2 t'$. Putting it all together, there is some t' such that $t \xrightarrow{(a_2,a_3)} t'$ and some x' such that $s' \sim_1 x' \sim_2 t'$ i.e., $s' \sim_1 \circ \sim_2 t'$.
   It follows that the transitive closure of any synchronous unwinding relation is a synchronous unwinding relation. In particular, the largest such relation must be transitive.
2. Let $\sim$ be the largest synchronous unwinding relation. By definition and the Item 1 above, we already have that $\sim$ is symmetric and transitive, so it suffices to show reflexivity. Let s be a reachable state. In this case there is a run $s_0 a_1 s_1 \dots a_n s_n$ of M such that $s = s_n$. We need to show that $s \sim s$. The proof is by induction on the length of the run.
   The base case of $s_0 \sim s_0$ is immediate from Item 1 of Definition 2.5. Suppose $s_i \sim s_i, 0 \le i \le n$. Assume $s_n \xrightarrow{a_{n+1}} s_{n+1}$. As $s_n \sim s_n$, and applying Item 3 of Definition 2.5 to the right handside copy of s_n, there exists a transition $s_n \xrightarrow{a_{n+1}} s'$ for some s' and $s_{i+1} \sim s'$. Because $\sim$ is symmetric, we have $s' \sim s_{n+1}$, and by transitivity $s_{n+1} \sim s_{n+1}$.
3. Any synchronous unwinding relation is still a synchronous unwinding relation when restricted to the reachable states. Conversely, given a synchronous unwinding relation on the reachable states, the (identical) relation which extends this to all states by union with the empty relation on unreachable states is also a synchronous unwinding relation. □

Using Lemma 6, we can design an algorithm to compute the largest synchronous unwinding relation, or the empty relation if none exists. By part (3) of Lemma 6 we may assume that all the states of the machine $M = \langle S, A, s_0, \to, O, obs \rangle$ are reachable.

The algorithm is an adaptation of the algorithm for calculating the relational coarsest partition by Kanellakis and Smolka [28]. For a partition P or equivalence relation $\approx$, we write $[s]_P$ or $[s]_\approx$ for the equivalence class containing element s. We say that a partition P of the state space S is *stable* if the corresponding equivalence relation $\sim_P$ satisfies

Algorithm 1: COMPUTE-LARGEST-UNWINDING(M).

Input: A synchronous machine $M = \langle S, A, s_0, \rightarrow, O, obs\rangle$. /* We assume S is reachable */
Output: The largest unwinding partition P on M, or $\varnothing$ if none exists
/* Initial partition P is given by the set of L observations */
$P \leftarrow \{obs_L^{-1}(o) \mid o \in O\}$;
Loop: /* Check stability of each subset $p \in P$ */
foreach $p \in P$ **do**
  foreach $s \in p$ *and* $(a_H, a_L) \in A$ **do**
    Let $R(s, a_H, a_L) = \{[t]_P \mid s \xrightarrow{(a_H, a_L)} t\}$;
  /* Check reflexive case of condition (3) */
  foreach $s \in p$ *and* $a_3 \in A_L$ **do**
    if *there exists* $a_1, a_2 \in A_H$ *with* $R(s, a_1, a_3) \neq R(s, a_2, a_3)$ **then**
      return $\varnothing$;
  /* $R(s, a_H, a_L)$ is independent of a_H */
  Fix $a_H \in A_H$;
  For $s, t \in p$ and $a_3 \in A_L$ let $s \approx_{a_3} t$ when $R(s, a_H, a_3) = R(t, a_H, a_3)$;
  if *there exists* $s, t \in p$ *and* $a_3 \in A_L$ *with* $s \neq t$ *and* $s \not\approx_{a_3} t$ **then**
    /* split p according to equivalence classes $[s]_{\approx_{a_3}}$ of $\approx_{a_3}$ */
    $P \leftarrow (P \setminus \{p\}) \cup \{[s]_{\approx_{a_3}} \mid s \in p\}$;
    Go to Loop
/* P contains the largest stable partition */
return P;

condition (3) of Definition 2.5.[3] The idea of the algorithm is to compute the coarsest stable partition satisfying condition (2) of Definition 2.5, by iteratively refining an existing partition if the latter is not stable. Given the current partition P, if there exists a (reachable) state s such that condition (3) of Definition 2.5 is not satisfied with $t = s$, the algorithm terminates and returns the empty relation: this follows from Lemma 6(2) because reflexivity is a necessary condition for the existence of an unwinding relation. Otherwise, we check whether it is stable for $s \neq t$. If it is, we have found the largest unwinding relation. If not, we refine the current partition based on the counterexample found.

The procedure is given by Algorithm 1. In each refinement step, with the current partition equal to P, we first compute the set

$$R(s, a_H, a_L) = \{[t]_P \mid s \xrightarrow{(a_L, a_H)} t\}$$

for each state s, and L action a_L and H action a_H. Rule (3) with respect to $\sim_P$ is equivalent to the statement that if $s \sim_P t$ then $R(s, a_H, a_L) = R(t, a'_H, a_L)$ for all L actions a_L and H actions a_H, a'_H. We first check this when $s = t$. Note that once this condition has been verified, we have verified that $R(s, a_H, a_L)$ does not depend on the second argument a_H. To check the non-reflexive cases, it therefore suffices to check the condition with respect to any fixed a_H.

We can now prove Theorem 6:

Proof of Theorem 6. Algorithm 1 terminates when no split occurs in the main loop. Since, when a split occurs, the new partition is a strict refinement of the previous one, the number of iterations of the main loop is at most $|S|$. For each $p \in P$, computing the function R can be done in time $O((|A_H| \times |A_L| \times |p|) + |p \cdot \rightarrow|)$. Checking the reflexive cases can be done in time $O(|A_H|^2 \times |A_L| \times |p|)$, and the non-reflexive cases can be done in time $O(|p|^2 \times |A_L|)$. Hence the **foreach** loop over $p \in P$ can be handled in time $O(|A_H|^2 \times |A_L| \times |S|^2 + |\rightarrow|)$. Since there are at most $|S|$ iterations, we have a total time of $O(|A_H|^2 \times |A_L| \times |S|^3 + |\rightarrow| \times |S|)$. Because $|\rightarrow|$ may be as large as $|S|^2$, this is $O(|A_H|^2 \times |A_L| \times |S|^3)$. (Literature subsequent to Kanellakis and Smolka has shown how to optimize their algorithm using careful scheduling, union-find data structures and amortized analysis, as well as parallel implementation. Similar optimizations may be applicable to our algorithm, but we will not pursue this here.)

To argue correctness, we first show that if there exists a synchronous unwinding $\sim$ on M, corresponding to partition $P_\sim$, the algorithm maintains the invariant that $P_\sim$ is a refinement of P. That this holds for the initial value of P follows from condition (2) of Definition 2.5. The only case where P changes value is where we have $p \in P$ and $a_3 \in A_L$ with

1. $R(s, a_1, a_3) = R(s, a_2, a_3)$ for all $s \in p$ and $a_1, a_2 \in A_H$, (there are no reflexivity violations), and
2. $R(s, a_H, a_3) \neq R(t, a_H, a_3)$ for some $s, t \in p$ and $a_H \in A_H$.

[3] Our definition of stability differs from that of Kanellakis and Smolka: they require that for each of a set of functions $f_a : S \rightarrow \mathcal{P}(S)$ (each corresponding to transitions with respect to some label a), and partition $p \in P$, for states $s, t \in S$ we have $s \sim_P t$ implies that that $f_a(s) \cap p = \varnothing$ iff $f_a(t) \cap p \neq \varnothing$. In this condition, we apply the same function to s and t. Our definition amounts to the application of different functions to s and t, corresponding to transitions with respect to (a_1, a_3) and (a_2, a_3), respectively.

In this case, we obtain the new value P' for P by splitting p into the collection $\{[s]_{\approx_{a_3}} \mid s \in p\}$, where $\approx_{a_3}$ is defined on p by $s \approx_{a_3} t$ if $R(s, a_H, a_3) = R(t, a_H, a_3)$. Suppose that $P_\sim$ is not a refinement of P'. Since the only element of P that changed was p, we must have $s, t \in p$ with $s \sim t$ and $R(s, a_H, a_3) \neq R(t, a_H, a_3)$. The latter means that there exists $p' \in P$ with (without loss of generality) $p' \in R(s, a_H, a_3)$ and $p' \notin R(t, a_H, a_3)$. That is, there exists $s' \in p'$ such that $s \xrightarrow{(a_H, a_3)} s'$, but for all t' with $t \xrightarrow{(a_H, a_3)} t'$ we have $t' \notin p'$. Because $\sim$ is a synchronous unwinding, $s \sim t$, and $s \xrightarrow{(a_H, a_3)} s'$, there exists t' with $t \xrightarrow{(a_H, a_3)} t'$ and $s' \sim t'$. But because $P_\sim$ refines P, this implies that $t' \in [s']_P = p'$, a contradiction. We conclude that in fact P' refines P.

The correctness argument now follows straightforwardly. Suppose that the algorithm outputs $\varnothing$: we show that there exists no synchronous unwinding on M. Suppose to the contrary that $\sim$ is a synchronous unwinding. At the time the algorithm terminates, we have $R(s, a_1, a_3) \neq R(s, a_2, a_3)$ for some $p \in P$, some $s \in p$ and some $a_1, a_2 \in A_H$. Without loss of generality, there exists some $p' \in P$ and $t \in S$ such that $s \xrightarrow{(a_1, a_3)} t \in p'$ but there exists no $t' \in p'$ such that $s \xrightarrow{(a_2, a_3)} t' \in p'$. By reflexivity of $\sim$, we have $s \sim s$, so there exists t' such that $s \xrightarrow{(a_2, a_3)} t'$ and $t \sim t'$. Because $P_\sim$ is a refinement of P, we obtain $t' \in [t]_P = p'$, a contradiction. We conclude that there exists no synchronous unwinding.

Conversely, suppose that the algorithm outputs a partition $P \neq \varnothing$, and let $\sim_P$ be the corresponding equivalence relation. Since P is a refinement of $\{obs_L^{-1}(o) \mid o \in O\}$, we have that $s \sim_P t$ implies $obs_L(s) = obs_L(t)$, so condition (2) of Definition 2.5 is satisfied. Moreover, we have, for all $p \in P$, that

1. $R(s, a_1, a_3) = R(s, a_2, a_3)$ for all $s \in p$ and $a_1, a_2 \in A_H$,
2. $R(s, a_H, a_3) = R(t, a_H, a_3)$ for all $s \neq t \in p$ and $a_H \in A_H$.

Together, these imply that $\sim_P$ satisfies condition (3) of Definition 2.5. Finally, since P is a partition, we have $s_0 \sim_P s_0$, so condition (1) also holds. □

6. Related work

In asynchronous machines the verification complexities of NDI and NDS are both PSPACE-complete, and RES (based on asynchronous unwinding) is in polynomial time [18,19,43]. Interestingly, PSPACE is also the complexity result for verifying Mantel's BSP conditions [30] on asynchronous finite state systems. For (asynchronous) push-down systems, the verification problem is undecidable [17]. The notion BNDC [18], essentially strengthens non-deducibility on strategies in an asynchronous process algebra setting by replacing trace equivalence by bisimulation equivalence. Two variants of BNDC, that restrict the quantification over the High attacker to finite or only "regularly divergent" processes, are shown in [33] to be decidable, by reduction to an EXPTIME problem, but exact complexity bounds are not provided. A general framework for asynchronous unwinding definitions is proposed in [7,8], and it is shown in [7] that the resulting security notions are PTIME decidable. This framework has been extended to a range of settings and applied to a variety of applications [11,15,21].

A number of works have defined notions of security for synchronous or timed systems, but fewer complexity results are known. Köpf and Basin [26] define a notion similar to RES and show it is PTIME decidable. Similar definitions are also used in the literature on language-based security [3,45].

Our semantic model is discrete-time with observers who see every clock tick. In the context of *dense-time systems* (defined by timed automata [2]) various notions of non-interference [18] have been defined [25] which state, using either trace-based or (bi)-simulation-based definitions that Low cannot distinguish the case where High does not act at all from the case where High may choose any action at any time. (These definitions consider just two cases of possible High strategy, whereas NDS quantifies over all High strategies, though there is the further difference that our framework requires High to act at all times it is scheduled.) A related problem, the *opacity* verification problem is studied in [12]. [25,12] show that these basic non-interference verification and opacity problems are undecidable for timed automata. More recent works [4,5] investigate subclasses of timed automata for which the non-interference problems become decidable along with solutions to corresponding controller synthesis problems. The restrictions required for decidability (e.g., determinism of transitions for a given action) are more severe than in our results for the discrete case.

Focardi et al. [20] define a spectrum of definitions related to ours in a timed process algebraic setting, and state a decidability result for one of them, close to our notion NDS. However, this result concerns an approximation to the notion "timed nondeducibility on compositions" (tBNDC) that is their real target, and they do not give a complexity result. Beauquier and Lanotte define *covert channels* in timed systems with *tick* transitions by using strategies [9]. They prove that the problem of the existence of a covert channel in such systems is decidable. However, their definition of covert channel requires that H and L have strategies to force a system into sets of runs with disjoint sets of L views. The induced definition on *free of covert channels* appears to be a weaker notion than NDS.

It has been noted in [7] that it is of interest to consider security properties that are *persistent* in the sense that a system is secure iff it remains secure when the initial state is reset to any reachable state. (This is motivated in [7] using arguments about process mobility, but it can also be understood as saying that even if all secrets generated to a given moment of time should leak, the system operates in such a way that secrets freshly generated from that time on are kept secret.) Of the properties we have considered, RES is persistent (this is immediate from Lemma 6), but NDI and NDS are not. (Fig. 6 gives

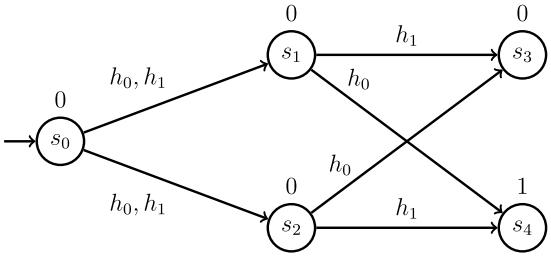

Fig. 6. A system non-persistently satisfying NDI and NDS.

an example of a system that satisfies both NDS and NDI from the initial state s_0, but not from the state s_1. Low actions do not affect the transitions, so are omitted.) Simply by enumerating all reachable states, our upper bounds on the complexity of NDI and NDS also apply to the persistent versions of these definitions, which check these properties from all reachable states. The lower bound of PSPACE for NDI also applies to the persistent version of NDI, because in the construction we gave for the lower bound, high actions have no effect after the initial state. We would conjecture that the persistent version of NDS is also EXPSPACE-hard, but leave this open.

For verification of large scale systems, it is desirable to have a compositional methodology, so that to prove security of a large system it suffices to prove security of its subcomponents and check that the composition has been done in a safe way. Compositionality of definitions of security has therefore been a topic of interest [10]. Our notion NDI is known to be non-compositional [34] but our notion of NDS is closed under composition [35], as are notions similar to our RES [26,34, 32]. These results require a slightly more fine-grained semantic model than the one that has sufficed for our complexity analysis in this paper, with inputs and outputs composed of multiple lines that can be hooked up in different ways to form a composition. We therefore do not attempt to state compositionality results in detail here. (We remark that in an asynchronous setting, the notion BNDC, which is similar in spirit to our NDS, is not compositional [33] but there exist stronger compositional variants [7].)

The definitions of security we have considered in this paper are based on *possibilistic* notions of information flow. The literature has also considered probabilistic notions of information flow. Wittbold and Johnson [46] showed that several possibilistic definitions of security, including NDI and NDS, may hold in a natural class of systems that nevertheless have probabilistic flows of information. Gray [23,22] defined probabilistic variants of NDI, NDS and RES in a discrete-time state machine model. More recently, related definitions have been proposed in contexts of process algebra [1] and dense-time automata [29]. There has been limited work on automated verification of such probabilistic notions. [13,16] develop complexity results for probabilistic deducibility of initial input values in terminating programs, and probabilistic opacity problems are considered in [6]. In general, probabilistic verification problems can be expected to have significantly higher computational complexity than corresponding possibilistic versions of these problems. This being the case, the possibilistic definitions remain of interest, as providing more efficiently testable sufficient conditions for detecting *insecurity* of a system.

7. Conclusion

We remarked above that nondeducibility-based notions of security may have the disadvantage that they do not readily support a compositional approach to secure systems development, motivating the introduction of unwinding-based definitions of security. The complexity results of the present paper can be interpreted as lending further support to the value of unwinding-based definitions. We have found that the two nondeducibility notions we have considered, while both decidable, are intractable. On the other hand, the unwinding-based notion of synchronous restrictiveness has tractable complexity. This makes this definition a more appropriate basis for automated verification of security. Even if the desired security property is nondeducibility on inputs or nondeducibility on strategies, it is sufficient to verify that a system satisfies synchronous restrictiveness, since this is a stronger notion of security. It remains to be seen whether there is a significant number of practical systems that are secure according to the nondeducibility-based notions, but for which there does not exist a synchronous unwinding. If so, then an alternate methodology needs to be applied for the verification of security for such systems.

References

[1] A. Aldini, M. Bravetti, R. Gorrieri, A process-algebraic approach for the analysis of probabilistic noninterference, J. Comput. Secur. 12 (2) (2004) 191–245.
[2] R. Alur, D.L. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.
[3] J. Agat, Transforming out timing leaks, in: Proc. ACM Symp. on Principles of Programming Languages, 2000, pp. 40–53.
[4] G. Benattar, F. Cassez, D. Lime, O.H. Roux, Synthesis of non-interferent timed systems, in: Proc. of the 7th Int. Conf. on Formal Modeling and Analysis of Timed Systems, FORMATS'09, in: Springer LNCS, vol. 5813, 2009, pp. 28–42.
[5] G. Benattar, F. Cassez, D. Lime, O.H. Roux, Control and synthesis of non-interferent timed systems, Internat. J. Control 88 (2) (2015) 217–236.
[6] B. Bérard, K. Chatterjee, N. Sznajder, Probabilistic opacity for Markov decision processes, Inform. Process. Lett. 115 (1) (2015) 52–59.
[7] A. Bossi, R. Focardi, C. Piazza, S. Rossi, Verifying persistent security properties, Comput. Lang. Syst. Struct. 30 (2004) 231–258.
[8] A. Bossi, R. Focardi, D. Macedonio, C. Piazza, S. Rossi, Unwinding in information flow security, Electron. Notes Theor. Comput. Sci. 99 (2004) 127–154.

[9] D. Beauquier, R. Lanotte, Hiding information in multi level security systems, in: Proc. 4th Int. Workshop on Formal Aspects in Security and Trust, in: Springer LNCS, vol. 4691, 2006, pp. 250–269.
[10] A. Bossi, C. Piazza, S. Rossi, Compositional information flow security for concurrent programs, J. Comput. Secur. 15 (3) (2007) 373–416.
[11] A. Bossi, D. Macedonio, C. Piazza, S. Rossi, Information flow in secure contexts, J. Comput. Secur. 13 (3) (2005) 391–422.
[12] F. Cassez, The dark side of timed opacity, in: Proc. 3rd Int. Conf. on Information Security and Assurance, ISA'09, in: Springer LNCS, vol. 5576, 2009, pp. 21–30.
[13] P. Cerný, K. Chatterjee, T.A. Henzinger, The complexity of quantitative information flow problems, in: Proc. Computer Security Foundation Symp, 2011, pp. 205–217.
[14] F. Cassez, R. van der Meyden, C. Zhang, The complexity of synchronous notions of information flow security, in: Proc. FOSSACS, in: Springer LNCS, vol. 6014, 2010, pp. 282–296.
[15] S. Crafa, S. Rossi, Controlling information release in the pi-calculus, Inform. and Comput. 205 (8) (2007) 1235–1273.
[16] R. Chadha, M. Ummels, The complexity of quantitative information flow in recursive programs, in: Proc. FSTTCS 2012, in: LIPIcs, vol. 18, 2012, pp. 534–545.
[17] D. D'Souza, R. Holla, J. Kulkarni, R.K. Ramesh, B. Sprick, On the decidability of model-checking information flow properties, in: Proc. Int. Conf. on Information Systems Security, 2008, pp. 26–40.
[18] R. Focardi, R. Gorrieri, A classification of security properties for process algebras, J. Comput. Secur. (1995) 5–33.
[19] R. Focardi, R. Gorrieri, The compositional security checker: a tool for the verification of information flow security properties, Technical report UBLCS-96-14, Università di Bologna, August 1996.
[20] R. Focardi, R. Gorrieri, F. Martinelli, Information flow analysis in a discrete-time process algebra, in: Proc. Computer Security Foundation Workshop, 2000, pp. 170–184.
[21] R. Focardi, S. Rossi, Information flow security in dynamic contexts, J. Comput. Secur. 14 (1) (2006) 65–110.
[22] J.W. Gray III, Toward a mathematical foundation for information flow security, in: Proc. IEEE Symp. on Security and Privacy, 1991, pp. 21–35.
[23] J.W. Gray III, Probabilistic interference, in: Proc. IEEE Symp. on Security and Privacy, 1990, pp. 170–179.
[24] J.A. Goguen, J. Meseguer, Unwinding and inference control, in: Proc. IEEE Symp. on Security and Privacy, 1984, pp. 75–87.
[25] G. Gardey, J. Mullins, O.H. Roux, Non-interference control synthesis for security timed automata, in: Proc. 3rd Int. Workshop on Security Issues in Concurrency, SecCo'05, 2005.
[26] B. Köpf, D.A. Basin, Timing-sensitive information flow analysis for synchronous systems, in: Proc. European Symp. on Research in Computer Security, in: Springer LNCS, vol. 4189, Springer, 2006, pp. 243–262.
[27] P.C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in: N. Koblitz (Ed.), CRYPTO, in: Springer LNCS, vol. 1109, 1996, pp. 104–113.
[28] P.C. Kanellakis, S.A. Smolka, CCS expressions, finite state processes, and three problems of equivalence, in: Proc. 2nd Annual ACM Symp. on Principles of Distributed Computing, New York, NY, 1983, pp. 228–240.
[29] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Information flow analysis for probabilistic timed automata, in: Proc. Int. Workshop on Formal Aspects in Security and Trust, 2004, pp. 13–26.
[30] H. Mantel, Possibilistic definitions of security – an assembly kit, in: Proc. Computer Security Foundations Workshop, 2000, pp. 185–199.
[31] H. Mantel, Unwinding security properties, in: Proc. European Symp. on Research in Computer Security, in: Springer LNCS, vol. 1895, 2000, pp. 238–254.
[32] H. Mantel, On the composition of secure systems, in: Proc. IEEE Symp. on Security and Privacy, 2002, pp. 88–101.
[33] F. Martinelli, Partial model checking and theorem proving for ensuring security properties, in: Proc. Computer Security Foundations Workshop, 1998, pp. 44–52.
[34] D. McCullough, Noninterference and the composability of security properties, in: Proc. IEEE Symp. on Security and Privacy, 1988, pp. 177–186.
[35] J.K. Millen, Hookup security for synchronous machines, in: Proc. Computer Security Foundations Workshop, 1990, pp. 84–90.
[36] C. Percival, Cache missing for fun and profit, in: Proc. BSDCan, 2005.
[37] J.H. Reif, The complexity of two-player games of incomplete information, J. Comput. System Sci. 29 (2) (1984) 274–301.
[38] J. Rushby, Noninterference, transitivity, and channel-control security policies, Technical report, SRI international, Dec 1992.
[39] L.J. Stockmeyer, A.K. Chandra, Provably difficult combinatorial games, SIAM J. Comput. 8 (2) (1979) 151–174.
[40] L.J. Stockmeyer, A.R. Meyer, Word problems requiring exponential time (preliminary report), in: Proc. ACM Symp. on Theory of Computing, 1973, pp. 1–9.
[41] D. Sutherland, A model of information, in: Proc. National Computer Security Conf, 1986, pp. 175–183.
[42] R. van der Meyden, C. Zhang, A comparison of semantic models for noninterference, Theoret. Comput. Sci. 411 (7) (2010) 4123–4147.
[43] R. van der Meyden, C. Zhang, Algorithmic verification on noninterference properties, in: ENTCS, vol. 168, 2007, pp. 61–75.
[44] R. van der Meyden, C. Zhang, Information flow in systems with schedulers, Part I: Definitions, Theoret. Comput. Sci. 467 (2013) 68–88.
[45] D.M. Volpano, G. Smith, A type-based approach to program security, in: TAPSOFT, in: Springer LNCS, vol. 1214, Springer, 1997, pp. 607–621.
[46] J.T. Wittbold, D.M. Johnson, Information flow in nondeterministic systems, in: Proc. IEEE Symp. on Security and Privacy, 1990, pp. 144–161.
[47] Z. Wang, R.B. Lee, Covert and side channels due to processor architecture, in: Proc. Annual Computer Security Applications Conf, 2006, pp. 473–482.