# DIGITAL ACCESS TO
# SCHOLARSHIP AT HARVARD

# Deriving Epistemic Conclusions from Agent Architecture

| | |
|---|---|
| Citation | Chong, Stephen and Ron van der Meyden. 2009. Deriving epistemic conclusions from agent architecture. In Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge(TARK): July 6-8, 2009, California, ed. A. Heifetz, 61-70. New York, NY: ACM Press. |
| Published Version | http://doi.acm.org/10.1145/1562814.1562826 |
| Accessed | February 18, 2015 7:31:56 AM EST |
| Citable Link | http://nrs.harvard.edu/urn-3:HUL.InstRepos:3637832 |
| Terms of Use | This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Open Access Policy Articles, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#OAP |

*(Article begins on next page)*

# Deriving Epistemic Conclusions from Agent Architecture

**Stephen Chong**
School of Engineering and Applied Sciences
Harvard University*

**Ron van der Meyden**
School of Computer Science and Engineering
University of New South Wales

## 1   Introduction

One of our most resilient intuitions is that causality is a precondition for information flow: where there are no causal connections, we expect there to be no flow of information. In this paper, we study this idea as it arises in the computer science notion of systems *architectures*, which are high level designs that describe the coarse structure of a system in terms of its high-level components and their permitted causal interactions.

The MILS (Multiple Independent Levels of Security and Safety) initiative [Alves-Foss et al., 2006; Vanfleet et al., 2005; Boettcher et al., 2008] of the US Air Force proposes to use architecture as a key part of the certification case for high-assurance systems. It is envisaged that complex systems will be constructed from a mix of trusted and untrusted components composed according to an architecture. Trusted components might be formally verified to satisfy their specifications. Untrusted components might include commercial, off the shelf (COTS) software that is too complex to verify and may be unreliable or even malicious. It is desired that global security and safety properties can be guaranteed as a result of the behavior of trusted components and the architectural structure of the system, regardless of the behavior of untrusted components.

Architectures have generally been represented by diagrams of boxes and arrows whose meaning lacks a rigorous foundation. In a previous paper [Chong and van der Meyden, 2009] we gave formal semantics to a type of architectural specification. In addition to describing the system's causal structure, our specification format can express restrictions on information permitted to flow between components. We showed that these architectural specifications support interesting examples of rigorous reasoning about security properties expressed in an epis-temic logic. However, that framework does not deal well with situations in which agents synchronously observe information. In some applications, e.g., auctions, such synchronous observations are critical to the desired security and "fairness" properties. Fairness of an auction requires lower bounds on information flow, such as "the results of the previous round are known to all bidders". However, our previous approach expresses only upper bounds on information flow.

In this paper, we extend our previous approach by developing a richer architectural specification format that can express that certain components in the system synchronously observe the state of other components. This provides a way to represent lower bounds on information flow. Our extended specifications remain highly abstract: neither the states nor the agents' actions need to be explicitly specified. We show by means of a number of examples that the extended architectural specification format can enforce properties expressed using epistemic logic that cannot be enforced by our earlier framework.

There have to date been few examples to justify that an architectural approach to the construction of secure systems can be given a formal foundation. This work contributes by making formal sense of architectural specifications and showing that the type of reasoning envisaged by MILS has the potential to establish security properties at early stages of the design process. While our immediate motivation stems from computer security, we believe our results will also be of interest in software engineering, multi-agent Artificial Intelligence and economic mechanism design.

The structure of the paper is as follows. We begin with some preliminaries: Section 2 describes the formal machine model in which we interpret architectures, and Section 3 introduces the epistemic logic we use to specify security properties. In Section 4 we recall the architectural specification format of Chong and van der Meyden [2009], and motivate the need for the extension intro-

---

duced in the present paper. We define this extension and its semantics in Section 5. This is followed by a number of sections that give applications of the extended architectural specification format: Section 6 deals with a bulletin board architecture, Section 7 concerns an architecture for auction systems, and Section 8 presents a general condition under which a group of agents is able to audit all flows of information from one part of the system to another. Some concluding remarks are made in Section 9.

## 2 Machine model

Intuitively, the architectural specifications we develop in this paper are interpreted in systems consisting of agents that have state-changing actions and an ability to make observations of the system state. Following terminology from the security literature, we refer to agents as *domains*; the motivation for this usage is that we may be interested in information flows between classes of agents (e.g., all agents cleared to read classified documents) in addition to single agents. Actions are assumed to be deterministic. We assume that domains operate asynchronously.

We formalize systems using a *state-observed* machine model [Rushby, 1992], which defines deterministic state-based machines. A machine has a set of actions $A$, and each action is associated with a security domain. Intuitively, if action $a$ is associated with domain $u$, then $a$ represents a decision, choice, or action taken by the system component represented by $u$. Actions deterministically alter the machine state, and we assume that the observations of each security domain are determined by the current machine state.

Formally, a machine is a tuple $M = \langle S, s_0, A, \mathtt{step}, \mathtt{obs}, \mathtt{dom} \rangle$ where $S$ is a set of states, $s_0 \in S$ is the *initial state*, $A$ is a set of *actions*, $\mathtt{step} : S \times A \to S$ is a deterministic transition relation, $\mathtt{dom} : A \to D$ associates a domain with each action, and observation function $\mathtt{obs} : D \times S \to O$ describes for each state what observations can be made by each domain, for some set of observations $O$.

We assume that it is possible to execute any action in any state: function $\mathtt{step}$ is total. Given sequence of actions $\alpha \in A^*$, we write $s \cdot \alpha$ for the state reached by performing the each action in turn, starting in state $s$. We define $s \cdot \alpha$ inductively using the transition function $\mathtt{step}$, by ($\epsilon$ denotes the empty sequence)

$$s \cdot \epsilon = s \, ,$$
$$s \cdot \alpha a = \mathtt{step}(s \cdot \alpha, a) \, .$$

For notational convenience, we write $\mathtt{obs}_u$ for the function $\mathtt{obs}(u, \cdot)$, and $\mathtt{obs}_u(\alpha)$ for $\mathtt{obs}_u(s_0 \cdot \alpha)$, where $\alpha \in A^*$. If $G \subseteq D$ is a group of agents, we write $\mathtt{obs}_G(\alpha)$ for the tuple $(\mathtt{obs}_u(\alpha))_{u \in G}$. We also write $A_u = \{a \in A \mid \mathtt{dom}(a) = u\}$ for the set of actions belonging to $u \in D$.

Given a sequence $\alpha \in A^*$, the *view* of $\alpha$ of a group of domains $G$ is the sequence of $G$'s observations and actions that belong to $G$. The function $\mathtt{view}_G$ with domain $A^*$ is defined inductively by:

$$\mathtt{view}_G(\epsilon) = \mathtt{obs}_G(s_0) \, ,$$
$$\mathtt{view}_G(\alpha a) =$$
$$\begin{cases} \mathtt{view}_G(\alpha) \, a \, \mathtt{obs}_G(s_0 \cdot \alpha a) & \text{if } \mathtt{dom}(a) \in G \, , \\ \mathtt{view}_G(\alpha) \circ \mathtt{obs}_G(s_0 \cdot \alpha a) & \text{otherwise} \, . \end{cases}$$

The definition uses the absorptive concatenation operator $\circ$: for set $X$, sequence $\alpha \in X^*$, and element $x \in X$, $\alpha \circ x = \alpha$ if $x$ is equal to the last element of $\alpha$, and $\alpha \circ x = \alpha x$ otherwise. Intuitively, $G$'s view is the information that the group $G$ would have if it were acting as a single agent operating with asynchronous perfect recall, remembering all its distinct observations and actions, but not being aware of the duration of invariant observations. When $G = \{u\}$ is a singleton, we write $\mathtt{view}_u$ for $\mathtt{view}_G$. When $G$ is not a singleton, $\mathtt{view}_G(\alpha)$ can be understood as the maximal information that the group could have if we were to give it the ability to instantaneously communicate within the group any new information that any agent in the group acquires.

Finally, for any sequence of actions $\alpha \in A^*$, we write $\alpha \upharpoonright G$ for the subsequence of $\alpha$ of actions whose domain is in the set $G$.

## 3 Specification Language

We use an epistemic logic to express information flow properties of architectures, with grammar:

$$\phi, \psi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \mathcal{K}_G \phi$$
$$\mathcal{K} ::= K \mid D \mid E \mid C$$
$$G, H \text{ range over groups of domains}$$

Formulas $\top$, $p$, $\neg\phi$, and $\phi \wedge \psi$ are standard from propositional logic: $\top$ is always satisfied, $p$ is a propositional constant, $\neg\phi$ is satisfied if $\phi$ is not satisfied, and $\phi \wedge \psi$ is satisfied if both $\phi$ and $\psi$ are satisfied. Epistemic formula $\mathcal{K}_G \phi$ is satisfied if the group of domains $G$ have knowledge of type $\mathcal{K}$ of formula $\phi$, where $\mathcal{K}$ ranges over $K, D, E$, and $C$. The formula $D_G \phi$ says $G$ has distributed knowledge of $\phi$; $E_G \phi$ says every domain in $G$ knows $\phi$; and $C_G \phi$ says $G$ has common knowledge

of $\phi$. The operator $K_G$ represents a novel distributed-knowledge-like notion that we call *group* knowledge.

Formulas are interpreted using a possible worlds semantics, where a world is a sequence of actions $\alpha \in A^*$. To interpret epistemic formulas, we require appropriate indistinguishability relations for each type of epistemic operator. Two sequences of actions $\alpha \in A^*$ and $\alpha' \in A^*$ are indistinguishable to a group of domains $G$, written $\alpha \approx_G \alpha'$, if $G$'s views of the two sequences are identical: $\alpha \approx_G \alpha' \iff \texttt{view}_G(\alpha) = \texttt{view}_G(\alpha')$. If $G = \{u\}$ we write $\approx_u$ for $\approx_G$: this is a standard notion of asynchronous perfect recall indistinguishability for a single agent.

As usual, we define indistinguishability relations $\approx_G^{\mathcal{K}}$ over action sequences, for group $G \subseteq D$ and knowledge kinds $\mathcal{K} \in \{K, D, E, C\}$: $\approx_G^K$ is $\approx_G$; $\approx_G^D$ is the intersection of $\approx_u$ for all $u \in G$; $\approx_G^E$ is the union of $\approx_u$ for all $u \in G$; and $\approx_G^C$ is the reflexive transitive closure of $\approx_G^E$. In addition to the standard abbreviations, we write $K_u\phi$ as shorthand for $K_{\{u\}}\phi$; this is the same as $E_{\{u\}}\phi$, $D_{\{u\}}\phi$ and $C_{\{u\}}\phi$.

A *proposition* is a set $X \subseteq A^*$. We say proposition $X$ is *non-trivial* if $X \neq \emptyset$ and $X \neq A^*$. An *interpretation function* $\pi$ is a function from propositional constants to propositions.

We define the semantics of the logic using satisfaction relation $M, \pi, \alpha \vDash \phi$, which intuitively means that formula $\phi$ is true given interpretation function $\pi$, and machine $M$ that has executed sequence $\alpha \in A^*$. Figure 1 defines satisfaction relation $M, \pi, \alpha \vDash \phi$. We write $M, \pi \vDash \phi$ if for all $\alpha \in A^*$ we have $M, \pi, \alpha \vDash \phi$.

To reason about information that is locally known to a group of domains $G$, we introduce $G$-local propositions [Engelhardt et al., 1998], and $G$-action-local propositions. A proposition is $G$-local if the view of domains $G$ suffices to decide the proposition. Formally, a set of states $X$ is a *$G$-local proposition* if for all $\alpha, \alpha' \in A^*$, if $\alpha \approx_G \alpha'$, then $\alpha \in X \iff \alpha' \in X$. A proposition is $G$-action-local if the actions of domains in $G$ suffice to decide the proposition. Formally, a set $X \subseteq A^*$ is a *$G$-action-local proposition* if for all $\alpha, \alpha' \in A^*$ if $\alpha \restriction G = \alpha' \restriction G$, then $\alpha \in X \iff \alpha' \in X$. Note that any $G$-action-local proposition is a $G$-local proposition, but not vice-versa. As usual, we abbreviate $\{u\}$ to $u$ in these notions.

## 4 Architectures

In this section we recall the notion of architecture and its semantics as introduced in Chong and van der Meyden [2009] and give an example that motivates the need for the extension that we develop in the present paper.

Architectures are specifications of the causal structure of a system that place constraints on the permitted flows of information between security domains. Figure 2 gives an example of an architecture for an electronic election system, comprised of a number of voters $v_1 \ldots v_n$, and an election authority $ElecAuth$.
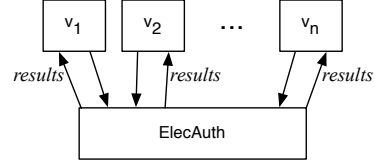


Figure 2: An architecture for elections

Intuitively, this diagram represents that the voters submit information (their votes) to the election authority, who returns the result of the election to the voters. In addition to these permitted information flows, the diagram also implies that some flows of information are prohibited. In particular, since there is no edge from a voter to any other voter, direct communication between voters through the election system is not permitted. Further, the election authority is not permitted to return more information to any voter than the results. In particular, the details of any individual voter's vote should not be revealed, except to the extent that information about it might be deducible from the result of the election.

Formally, an architecture is a pair $\mathcal{A} = (D, \rightarrowtail)$, where $D$ is a set of security domains, and $\rightarrowtail \subseteq D \times D \times (\mathcal{L} \cup \{\top\})$, where $\mathcal{L}$ is a set of function names. We write $u \xrightarrow{f} v$ when $(u, v, f) \in \rightarrowtail$, and write $u \rightarrowtail v$ as shorthand for $\exists f. (u, v, f) \in \rightarrowtail$, and $u \not\rightarrowtail v$ as shorthand for $\neg\exists f. (u, v, f) \in \rightarrowtail$.

The intuition for the relation $u \xrightarrow{f} v$ is that information flow from $u$ to $v$ is permitted, but may be subject to constraints. In case $f = \top$, there are no constraints on information flow from $u$ to $v$: any information that may be possessed by $u$ is permitted to be passed to $v$ when $u$ acts. If $f \in \mathcal{L}$ then information is allowed to flow from domain $u$ to domain $v$, but it needs to be *filtered* through the function denoted by $f$: only information output by this function may be transmitted from $u$ to $v$. If $u \not\rightarrowtail v$ then no direct flow of information from $u$ to $v$ is permitted.

Architectures are required to have the following properties:

**Arch1.** For all $u, v \in D$, there exists at most one $f \in \mathcal{L} \cup \{\top\}$ such that $u \xrightarrow{f} v$.

3

$$M, \pi, \alpha \vDash \top$$

$$M, \pi, \alpha \vDash p \quad \text{iff } \alpha \in \pi(p)$$

$$M, \pi, \alpha \vDash \neg\phi \quad \text{iff } M, \pi, \alpha \nvDash \phi$$

$$M, \pi, \alpha \vDash \phi \wedge \psi \quad \text{iff } M, \pi, \alpha \vDash \phi \text{ and } M, \pi, \alpha \vDash \psi$$

$$M, \pi, \alpha \vDash \mathcal{K}_G\phi \quad \text{iff } M, \pi, \alpha' \vDash \phi \text{ for all}$$

$$\alpha' \in A^* \text{ s.t. } \alpha \approx_G^{\mathcal{K}} \alpha'$$

Figure 1: $M, \pi, \alpha \vDash \phi$

**Arch2.** The relation $\rightarrowtail$ is reflexive in that for all $u \in D$ we have $(u, u, \top) \in \rightarrowtail$.

The first condition requires that all permitted flows of information from $u$ to $v$ are represented using a single labeled edge. Intuitively, any policy with multiple such edges can always be transformed into one satisfying this condition, by combining the pieces of information flowing across these edges into a tuple that flows across a single edge. The second condition is motivated from the fact that information flow from a domain to itself cannot be prevented. When drawing architectures, we annotate arrows between domains with the filter function names and elide reflexive arrows. For arrows drawn without a label, and elided reflexive arrows, the implied label is $\top$.

Architectures do not define the interpretations of the function names $\mathcal{L}$. If $\mathcal{A} = (D, \rightarrowtail)$ is an architecture, an *interpretation* for $\mathcal{A}$ is a tuple $\mathcal{I} = (A, \mathtt{dom}, \mathtt{I})$, where $A$ is a set of actions, $\mathtt{dom} : A \to D$ assigns these actions to domains of $\mathcal{A}$, and $\mathtt{I}$ is a function mapping each $f \in \mathcal{L}$ to a function with domain $A^*$ (and arbitrary codomain). We call the pair $(\mathcal{A}, \mathcal{I})$ an *interpreted architecture*.

Intuitively, if $u \xrightarrow{f} v$ and $\alpha \in A^*$ and $a \in A$ is an action with $\mathtt{dom}(a) = u$, then $\mathtt{I}(f)(\alpha a)$ is the information that is permitted to flow from $u$ to $v$ when the action $a$ is performed after occurrence of the sequence of actions $\alpha$. Based on this intuition, we may define a function $\mathtt{fta}_u$ with domain $A^*$ that captures that maximal information that domain $u$ is permitted to have after a sequence of actions has been executed.

Given extended architecture $(D, \rightarrowtail)$ and an architectural interpretation $\mathcal{I} = (A, \mathtt{dom}, \mathtt{I})$, the function $\mathtt{fta}_u$ is defined inductively by $\mathtt{fta}_u(\epsilon) = \epsilon$, and, for $\alpha \in A^*$ and $a \in A$,

$$\mathtt{fta}_u(\alpha a) = \begin{cases} \mathtt{fta}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrow u \\ \mathtt{fta}_u(\alpha)(\mathtt{fta}_{\mathtt{dom}(a)}(\alpha), a) & \text{if } \mathtt{dom}(a) \xrightarrow{\top} u \\ \mathtt{fta}_u(\alpha)\mathtt{I}(f)(\alpha a) & \text{if } \mathtt{dom}(a) \xrightarrow{f} u \end{cases}$$

The intuition for this definition is as follows. First, the value $\mathtt{fta}_u(\alpha)$ is to be understood as a concrete representation of the maximal information that domain $u$ is permitted to have after the sequence of actions $\alpha$ has

been performed. The recursive definition describes how this maximal information grows when the action $a$ is performed after $\alpha$. This depends on the nature of the edge, if any, from $\mathtt{dom}(a)$ to $u$. The first clause says that if there is no edge from $\mathtt{dom}(a)$ to $u$, then $u$ is not permitted to learn anything when $a$ is performed. That is, we have $\mathtt{fta}_u(\alpha a) = \mathtt{fta}_u(\alpha)$, so $u$'s maximal information before and after the action is the same. The second clause says that if $\mathtt{dom}(a) \xrightarrow{\top} u$, then the additional information that $u$ may have after $a$ is the maximal information that the domain $\mathtt{dom}(a)$ performing the action had before $a$ (i.e. $\mathtt{fta}_{\mathtt{dom}(a)}(\alpha)$), together with the fact that the action $a$ has just been performed. Finally, in the case of edges of the form $\mathtt{dom}(a) \xrightarrow{f} u$, the interpretation of the function name $f$ defines the additional information that $u$ may acquire when $a$ is performed.

A machine complies with an interpreted extended architecture if it has appropriate domains and actions, and for each domain $u$, what $u$ observes in state $s_0 \cdot \alpha$ is determined by $\mathtt{fta}_u(\alpha)$. We call such a machine *FTA-compliant*. ("FTA" is derived from *filtered transmission* of information about *actions*.)

**Definition 1 (FTA-compliant)** *A machine* $M = \langle S, s_0, A, D, \mathtt{step}, \mathtt{obs}, \mathtt{dom} \rangle$ *is FTA-compliant with an interpreted architecture* $(\mathcal{A}, \mathcal{I})$, *with* $\mathcal{A} = (D', \rightarrowtail)$ *and* $\mathcal{I} = (A', \mathtt{dom}', \mathtt{I})$, *if* $A = A'$, $D = D'$, $\mathtt{dom} = \mathtt{dom}'$ *and for all domains* $u \in D$ *and all* $\alpha, \alpha' \in A^*$, *if* $\mathtt{fta}_u(\alpha) = \mathtt{fta}_u(\alpha')$ *then* $\mathtt{obs}_u(\alpha) = \mathtt{obs}_u(\alpha')$.

It is shown in Chong and van der Meyden [2009] that this definition of architectures and their semantics, together with some restrictions on the permitted interpretations of the function names, provides a specification format that, although highly abstract, already suffices to ensure that certain interesting epistemic consequences hold in any system that complies with the specification. For example, it is shown that, under some mild assumptions, the election architecture enforces the property that voters' actions in the system are anonymous. More precisely, any proposition considered possible by any voter $v$ about other voters is also considered possible by $v$ if we permute the names of the other voters. The assumptions required are that any action available to a voter is also available to any other voter and that the *results* function

4

is invariant under permutation of the voters, so that the results depend only on what the votes were, not on which voter did what.

Note that this result holds very abstractly: the architecture leaves most aspects of the system's design unspecified, and places only some weak constraints on its structure and the behavior of the election authority. It says nothing about the local states maintained by the voters, and does not specify the nature of the votes (so these can be implemented as, e.g., a single vote for the preferred candidate, with voting in a sequence of rounds, or as a preference order on all candidates). There is also considerable flexibility in the details of the results function (this could be, e.g., the winner of the election, or the number of votes cast for each candidate.) Moreover, the architecture isolates the responsibility for the desired security property (anonymity) in a single "trusted" component, the election authority. Assuming that security mechanisms such as access control and physical communication links have been applied to constrain information flow to the edges in the architecture, to obtain this property it suffices to check that the election authority correctly computes the result, and ensures that only this information can flow to the voters. This illustrates the way that the MILS approach to secure systems construction proposes to build secure systems from a mix of trusted and untrusted components composed in the context of an architectural framework that enforces constraints on information flow.

One might expect, from the diagram for the election architecture, that certain other properties should hold. One property that is desirable in the application, and which might seem to be implied by the architecture, is that the results of the election will be common knowledge to the voters. To formalize this let $M$ be a system that is compliant with the election architecture, and let $\pi(p)$ be a proposition about the result of the election, e.g., "party P is the winner". Then we might expect that $M, \pi \models K_v p \Rightarrow C_V p$, where $v$ is any voter and $V$ is the set of all voters. Unfortunately, this does not follow. Suppose we construct the system $M$ so that the election authority has an action $a$ that posts the election result only to a location that is observable to $v$, but to no other voter. Then after $a$, if $p$ holds then voter $v$ will know that $p$, but the other voters will not, so it will not be common knowledge. Nevertheless, such a system is not inconsistent with the architecture. The definition of FTA-compliance states only an upper bound on information flow, whereas what the expected property requires is a lower bound. Although it may superficially appear that the architectures of Chong and van der Meyden [2009] are able to represent broadcast systems, they are not able to enforce that broadcast information will be available

to intended recipients. One of the contributions of the present paper is to develop an extended notion of architecture that is able to capture such requirements.

## 5 Architectures with Observations

In order to express requirements such as the broadcast property expected in the election example discussed in the previous section, we now develop an extended notion of architecture that captures both upper and lower bounds on information flow. One of the well-known mechanisms by which agents obtain common knowledge is by simultaneous direct observation of their environment. Our extended architectural format is obtained by adding a new type of edge that represents observability of information.

An *architecture with filter functions and direct observations* (henceforth, simply *architecture*) is a tuple $\mathcal{A} = (D, \rightarrowtail, \dashrightarrow)$, where

1. $D$ is a set of domains,

2. $\rightarrowtail$ is a subset of $D \times D \times (\mathcal{L} \cup \{\top\})$, where $\mathcal{L}$ is a set of function names not including the special symbol $\top$, and

3. $\dashrightarrow \subseteq D \times D$.

We define architectural interpretations $\mathcal{I}$ and interpreted architectures $(\mathcal{A}, \mathcal{I})$ exactly as in the previous section (except that now $\mathcal{A}$ also contains a relation $\dashrightarrow$.)

This definition extends the definition of Chong and van der Meyden [2009] discussed in the previous section by adding edges of the form $u \dashrightarrow v$ to express that domain $v$ can observe domain $u$. More specifically, domain $v$ is required to observe the observations of domain $u$, in the sense that for all states $s$ and $t$, if $u \dashrightarrow v$ and $\mathsf{obs}_v(s) = \mathsf{obs}_v(t)$ then $\mathsf{obs}_u(s) = \mathsf{obs}_u(t)$. This imposes a lower bound on information flow in the system.

As in the previous section, we assume the properties **Arch1** and **Arch2** on the relation $\rightarrowtail$. Further, we assume that the relation $\dashrightarrow$ is transitive and acyclic (hence irreflexive). Note that the semantic explanation of $\dashrightarrow$ given above is transitive, so the assumption of transitivity of $\dashrightarrow$ is without loss of generality.[1] We write $u \overset{f}{\rightarrowtail} v$ if $(u, v, f) \in \rightarrowtail$, and write $u \dashrightarrow v$ if $(u, v) \in \dashrightarrow$. If

---

[1] We assume acyclicity of $\dashrightarrow$ for technical reasons: it may be possible to remove this assumption, but with significant complications to the definition of FTAO-compliance given below. However, if $u \dashrightarrow v$ and $v \dashrightarrow u$ then $\mathsf{obs}_u(s) = \mathsf{obs}_u(t)$ iff $\mathsf{obs}_v(s) = \mathsf{obs}_v(t)$, and so the effect of cycles can be obtained by defining $\mathsf{obs}_u = \mathsf{obs}_v$. Hence the loss of generality is not great.

there is no $f$ such that $u\overset{f}{\rightarrowtail}v$ then we write $u\not\rightarrowtail v$, and similarly write $u\not\dashrightarrow v$ if not $u\dashrightarrow v$.

We define the semantics of an architecture with filter functions and direct observation by defining a function $\mathtt{ftao}_u$, and requiring that $\mathtt{ftao}_u$ is the maximal information that $u$ is allowed to learn given a particular execution of the system. We also impose the lower bound on information flow mentioned above. Function $\mathtt{ftao}_u$ is similar to $\mathtt{fta}_u$, but (in order to be compatible with the lower bound) also incorporates the observations of domains $v$ such that $v\rightarrowtail u$. To define $\mathtt{ftao}_u$, we first define the *directly observable domains* $DO(u) = \{v \mid v\dashrightarrow u\}$ to be the set of domains that $u$ directly observes according to relation $\dashrightarrow$. Note that $u\notin DO(u)$. Also, we define an operator $+$ such that $x + y = \epsilon$ if $x = y = \epsilon$, and $x + y = (x, y)$ otherwise. For any sequence sequence $\sigma$ we treat $\sigma\epsilon$ as identical to $\sigma$.

Given architecture $(D, \rightarrowtail, \dashrightarrow)$, interpretation $\mathcal{I} = (A, \mathtt{dom}, \mathtt{I})$, and machine $M = \langle S, s_0, A, \mathtt{step}, \mathtt{obs}, \mathtt{dom}\rangle$, the function $\mathtt{ftao}_u$ is defined inductively by $\mathtt{ftao}_u(\epsilon) = \epsilon$, and, for $\alpha \in A^*$ and $a \in A$, $\mathtt{ftao}_u(\alpha a) = \mathtt{ftao}_u(\alpha)(x + y)$ where

$$x = \begin{cases} \epsilon & \text{if } \mathtt{dom}(a)\not\rightarrowtail u \\ (\mathtt{ftao}_{\mathtt{dom}(a)}(\alpha), a) & \text{if } \mathtt{dom}(a)\overset{\top}{\rightarrowtail}u \\ \mathtt{I}(f)(\alpha a) & \text{if } \mathtt{dom}(a)\overset{f}{\rightarrowtail}u \end{cases}$$

and

$$y = \begin{cases} \epsilon & \text{if } \mathtt{obs}_{DO(u)}(\alpha a) = \\ & \qquad\qquad \mathtt{obs}_{DO(u)}(\alpha) \\ \mathtt{obs}_{DO(u)}(\alpha a) & \text{otherwise.} \end{cases}$$

Intuitively, $\mathtt{ftao}_u(\alpha)$ represents the maximal information that domain $u$ is permitted to have acquired after the sequence of actions $\alpha$. The value $x + y$ represents the *new* information that $u$ is permitted to acquire when the action $a$ is performed after sequence $\alpha$. The component $x$ expresses that this includes the action $a$ plus any information that $\mathtt{dom}(a)$ may have, if $\mathtt{dom}(a)\overset{\top}{\rightarrowtail}u$, or that information filtered by the function $\mathtt{I}(f)$ if $\mathtt{dom}(a)\overset{f}{\rightarrowtail}u$. Moreover, the component $y$ expresses that domain $u$ is allowed to learn the observation of any domain in $DO(u)$, provided this differs from the previous observation. (As in the definition of $\mathtt{view}_u$, we reduce stuttering observations to a single copy to model asynchrony.)

The following definition gives semantics to interpreted architectures by stating that the information possessed by $u$ has an upper bound of $\mathtt{ftao}_u$ and a lower bound of the directly observable information. ("FTAO" is derived from *filtered transmission* of information about *actions* and *observations*.)

**Definition 2 (FTAO-compliance)** *Machine $M$ is FTAO-compliant with interpreted architecture $\mathcal{A}\mathcal{I} = (\mathcal{A}, \mathcal{I})$ where $\mathcal{A} = (D, \rightarrowtail, \dashrightarrow)$ and $\mathcal{I} = (A, \mathtt{dom}, \mathtt{I})$ if it has the same components $D$, $A$ and $\mathtt{dom}$, and*

1. *for all agents $u \in D$ and all $\alpha, \alpha' \in A^*$, if $\mathtt{ftao}_u(\alpha) = \mathtt{ftao}_u(\alpha')$ then $\mathtt{obs}_u(\alpha) = \mathtt{obs}_u(\alpha')$, and*

2. *for all agents $u, v \in D$ and all $s, t \in S$ if $u \dashrightarrow v$ and $\mathtt{obs}_v(s) = \mathtt{obs}_v(t)$ then $\mathtt{obs}_u(s) = \mathtt{obs}_u(t)$.*

In the following sections, we give some examples of architectural specifications based on architectures with filter functions and direct observations, and show that these specifications suffice to establish interesting epistemic specifications.

## 6 Example: Bulletin board

We begin with an example that shows that architectures with observations support the intuitions concerning broadcast discussed in Section 4.
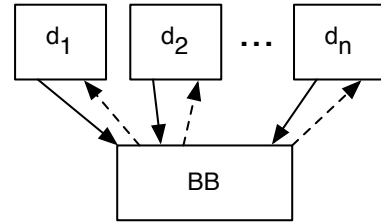


Figure 3: Architecture $\mathcal{B}\mathcal{B}$

Figure 3 shows architecture $\mathcal{B}\mathcal{B}$, a bulletin board architecture for $n$ agents. Each agent can send information to bulletin board $BB$, and directly observe $BB$. Thus, the architecture enforces that communication between agents must occur via the bulletin board. The bulletin board is a form of broadcast communication: announcements by one agent are observable by all agents.

Given two additional local assumptions about the behavior of the bulletin board, we can show that for distinct agents $c$ and $d$, any $c$-local proposition that is known to $d$ must also be known to everyone. Thus, the bulletin board architecture ensures that no agent has an unfair advantage over another agent: anything $d$ can learn about $c$ can also be learned by any other agent.

The first local assumption, BB1, guarantees that broadcast is not anonymous: any update to the bulletin board reveals the domain of the action that updated it.

**BB1.** For all $\alpha, \beta \in A^*$ and $a, b \in A$, if $\texttt{obs}_{BB}(\alpha) = \texttt{obs}_{BB}(\beta) \neq \texttt{obs}_{BB}(\beta b) = \texttt{obs}_{BB}(\alpha a)$ then $\texttt{dom}(a) = \texttt{dom}(b)$.

The second local assumption, BB2, ensures that the observable state of the bulletin board is a function of the previous observations of the bulletin board, and the current view of the poster. To state BB2, we first define $seq(f, \alpha)$, which, for any function $f$ with domain $A^*$, and any $\alpha \in A^*$, is the sequence of output of $f$ on successively longer prefixes of $\alpha$, with stuttering removed: $seq(f, \epsilon) = f(\epsilon)$ and $seq(f, \alpha a) = seq(f, \alpha) \circ f(\alpha a)$.

**BB2.** For all $\alpha, \beta \in A^*$ and $a \in A$, if $seq(\texttt{obs}_{BB}, \alpha) = seq(\texttt{obs}_{BB}, \beta)$ and $\texttt{view}_{\texttt{dom}(a)}(\alpha) = \texttt{view}_{\texttt{dom}(a)}(\beta)$ then $\texttt{obs}_{BB}(\alpha a) = \texttt{obs}_{BB}(\beta a)$.

The following theorem states that any bulletin board implementation $M$ that satisfies the bulletin board architecture $\mathcal{BB}$, and satisfies conditions BB1 and BB2, will not give any agent an unfair advantage over another: anything agent $d$ can learn about agent $c$ can be learned by everyone.

**Theorem 1** *Let $M$ be FTAO-compliant with interpreted architecture $\mathcal{AI} = (\mathcal{BB}, \mathcal{I})$ where $\mathcal{I} = (A, \texttt{dom}, \texttt{I})$, and satisfy conditions BB1 and BB2. For distinct agents $c, d \in D - \{BB\}$, if $\pi(p)$ is a c-local proposition, then $M, \pi \vDash K_d p \Rightarrow E_D p$.*

Indeed, we can show that if agent $c$ learns something about agent $d$, then it is common knowledge among all agents.

**Theorem 2** *Let $M$ be FTAO-compliant with interpreted architecture $\mathcal{AI} = (\mathcal{BB}, \mathcal{I})$ where $\mathcal{I} = (A, \texttt{dom}, \texttt{I})$, and satisfy conditions BB1 and BB2. Let $G = D - \{BB\}$. For distinct agents $c, d \in G$, if $\pi(p)$ is a c-local proposition, then $M, \pi \vDash K_d p \Rightarrow C_G p$.*

## 7 Example: Auction

Figure 4 shows architecture $\mathcal{AUC}$, an auction architecture for $n$ bidders, auctioneer $Auc$, and bulletin board $BB$. As in bulletin board architecture $\mathcal{BB}$, bidders directly observe bulletin board $BB$. Unlike $\mathcal{BB}$, bidders do not send information directly to the bulletin board, but instead submit bids to the trusted auctioneer, who posts auction results to the bulletin board. Information sent by the auctioneer to the bulletin board is bounded by the filter function denoted by $results$.

Intuitively, the auctioneer $Auc$ and bulletin board $BB$ are the trusted components of the system. We specify
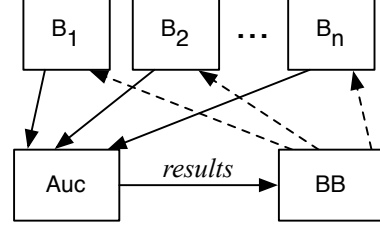


Figure 4: Architecture $\mathcal{AUC}$

additional local constraints for these components. Their compliance with the following constraints might be assured by means of a careful verification of their implementations.

We refer to information sent from the auctioneer to the bulletin board as an *announcement*. We restrict our attention to interpretations $\mathcal{I} = (A, \texttt{dom}, \texttt{I})$ for $\mathcal{AUC}$ that contain a distinguished announcement action $\text{ANN} \in A_{Auc}$. Intuitively, ANN is the only action that can send information to the bulletin board. According to the architecture, the maximal information for an announcement is the result of the filter function $\texttt{I}(results)$. We assume that this function has the following properties:

**A1.** Only the ANN action may send information to the bulletin board. If $a \neq \text{ANN}$ then $\texttt{I}(results)(\alpha a) = \epsilon$.

**A2.** Announcements depend only on the previous announcement, the individual behaviors of the bidders since the previous announcement (but not on their interleaving) and the action used to make the announcement. Let $\alpha, \alpha' \in A^*$, and let $\beta, \beta' \in A - \{\text{ANN}\}$. If $\texttt{I}(results)(\alpha\text{ANN}) = \texttt{I}(results)(\alpha'\text{ANN})$ and $\beta \upharpoonright B_i = \beta' \upharpoonright B_i$ for all $i = 1 \ldots n$, then $\texttt{I}(results)(\alpha\text{ANN}\beta\text{ANN}) = \texttt{I}(results)(\alpha'\text{ANN}\beta'\text{ANN})$.

**A3.** Announcements have the following *compensation* property: Let $\alpha \in A^*$, and let $\beta, \beta' \in A - \{\text{ANN}\}$. Let $B_i, B_j$ and $B_k$ be distinct bidders. If $\texttt{I}(results)(\alpha\text{ANN}\beta\text{ANN}) = \texttt{I}(results)(\alpha\text{ANN}\beta'\text{ANN})$ and $\beta \upharpoonright B_k = \beta' \upharpoonright B_k$ then there exists sequence $\beta'' \in (A_{B_1} \cup \ldots \cup A_{B_n})^*$ such that $\beta'' \upharpoonright B_i = \beta \upharpoonright B_i$ and $\beta'' \upharpoonright B_j = \beta' \upharpoonright B_j$ and $\texttt{I}(results)(\alpha\text{ANN}\beta\text{ANN}) = \texttt{I}(results)(\alpha\text{ANN}\beta''\text{ANN}) = \texttt{I}(results)(\alpha\text{ANN}\beta'\text{ANN})$.

Constraints A1-A3 allows us to reason about the behavior of function $results$. Constraint A3 is the most complex, and it intuitively requires that any pair of behaviors of $B_i$ and $B_j$ that are each individually consistent with

a given behavior of $B_k$ and a particular announcement, are jointly consistent with that announcement. There are several possible definitions for the function *results* that satisfy the constraints, such as a function that returns the maximum bid submitted since the last announcement, or a function that returns a list of the identity and bids of all bidders in decreasing order.

We also place local constraints on the bulletin board.

**BB1.** Each auctioneer announcement affects the bulletin board. More precisely, for all $\alpha \in A^*$, we have $\text{obs}_{BB}(\alpha\text{ANN}) \neq \text{obs}_{BB}(\alpha)$.

**BB2.** The bulletin board contains at least the most recent announcement of the auctioneer. If $\text{obs}_{BB}(\alpha\text{ANN}) = \text{obs}_{BB}(\alpha'\text{ANN})$, then $\text{I}(results)(\alpha\text{ANN}) = \text{I}(results)(\alpha'\text{ANN})$.

Various implementations of the bulletin board are compatible with these constraints. For example, the bulletin board may display the complete history of the auctioneer's announcements, or only the latest announcement plus the sequence number of that announcement, or the last $k$ announcements plus their sequence numbers.

The auction architecture and the local constraints allow us to prove that in any implementation that satisfies the auction architecture and local constraints, no bidder gains an information advantage over other bidders: if bidder $B_j$ learns something about bidder $B_i$, then it is common knowledge among all bidders.

**Theorem 3** *Let machine $M$ be FTAO-compliant with interpreted architecture $\mathcal{AI} = (\mathcal{AUC}, \mathcal{I})$ where $\mathcal{I} = (A, \text{dom}, \text{I})$, and satisfy conditions A1–A3 and BB1–BB2. Let $B = \{B_1, \ldots, B_n\}$ be the set of all bidders. For distinct bidders $B_i$ and $B_j$ if $\pi(p)$ is a $B_i$-local proposition, then $M, \pi \vDash K_{B_j} p \Rightarrow C_B p$.*

If we make further assumptions about the homogeneity of bidders, and that auctioneer announcements do not reveal bidder identities, we can prove that $\mathcal{AUC}$ provides bidder anonymity.

**B1.** All bidders have the same set of actions. For any bidder $B_i$, the set of possible actions $A_{B_i}$ is $\{a^{B_i} \mid a \in A_B\}$, where $A_B$ is the set of bidder actions.

We define a *bidder permutation $P$* as a permutation over the set of bidders $B$. Since all bidders have the same set of actions, we can apply a permutation $P$ to sequence $\alpha$, written $P(\alpha)$. We define $P(\epsilon) = \epsilon$, $P(\alpha a) = P(\alpha)a$ if $\text{dom}(a) \notin B$, and $P(\alpha a^{B_i}) = P(\alpha)a^{P(B_i)}$ if $a \in A_B$. We apply permutation $P$ to proposition $X \subseteq A^*$ by applying $P$ to each sequence $\alpha \in X$: $P(X) = \{P(\alpha) \mid \alpha \in X\}$. Intuitively, if $X$ is a $u$-local proposition, for $u \in B$, then $P(X)$ is a $P(u)$-local proposition. Using bidder permutations, we can state the local constraint that announcements do not depend on (and thus do not reveal) bidder identities.

**A4.** Announcements have the following *identity-oblivious* property: For all bidder permutations $P$, and sequences $\alpha$, $\text{I}(results)(\alpha) = \text{I}(results)(P(\alpha))$.

Given these additional local constraints we can show that if $B_i$ believes that some proposition $X$ may be satisfied, then $B_i$ also believes that $P(X)$ may be satisfied, for bidder permutation $P$. For example, if Bill considers it possible that Bob bid \$10 and Bonnie bid \$20, then Bill considers it possible that Bonnie bid \$10 and Bob bid \$20.

**Theorem 4** *Let system $M$ be FTAO-compliant with interpreted architecture $\mathcal{AI} = (\mathcal{AUC}, \mathcal{I})$ where $\mathcal{I} = (A, \text{dom}, \text{I})$, and satisfy conditions A1–A4, BB1–BB2, and B1. Let $B_i$ be a bidder. For all bidder permutations $P$ such that $P(B_i) = B_i$, if $\pi'(p) = P(\pi(p))$, and $M, \pi, \alpha \vDash \neg K_{B_i} \neg p$ then $M, \pi', \alpha \vDash \neg K_{B_i} \neg p$.*

## 8 Example: Observational barriers

Our last example is a general result that expresses a sense in which a group of domains can jointly function as an auditor of all information flows from one group of domains to another.

Let $\mathcal{A} = (D, \rightarrowtail, \dashrightarrow)$ be an architecture, and let $U, V, G \subseteq D$. Say that $G$ is an *observational barrier* separating $U$ from $V$ in $\mathcal{A}$ if there exists a partition $(L, R)$ of $D$, such that $U \cup G \subseteq L$, and $V \subseteq R$, there are no edges from $R$ to $L$, and the only edges leading from $L$ to $R$ are observational edges leading from $G$. That is,

- for all domains $u \in R$ and $v \in L$, we have $u \not\rightarrowtail v$ and $u \not\dashrightarrow v$, and

- for all domains $u \in L$ and $v \in R$, we have $u \not\rightarrowtail v$, and if $u \dashrightarrow v$ then $u \in G$.

The following result states that if $G$ is an observational barrier separating $U$ from $V$, then if $V$ has information about $U$, then that information is also available to $G$. This makes precise the sense in which $G$ is able to audit information flowing from $U$ to $V$.

**Theorem 5** *Let $\mathcal{AI} = (\mathcal{A}, \mathcal{I})$ be an interpreted architecture and suppose that $G$ is an observational barrier*
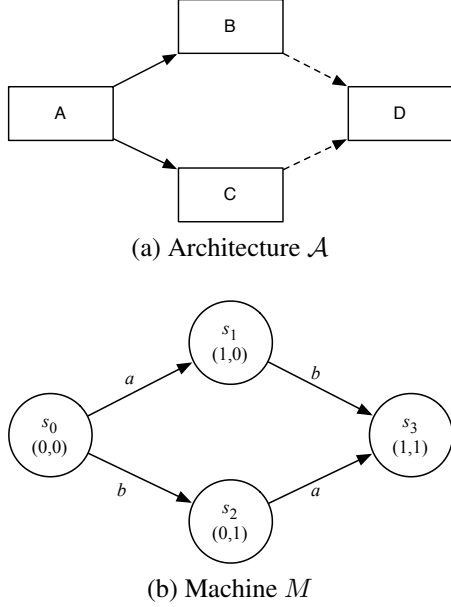
(a) Architecture $\mathcal{A}$



(b) Machine $M$

Figure 5: Observational barrier example

*separating $U$ from $V$ in $\mathcal{A}$. Let $M$ be FTAO-compliant with $\mathcal{AI}$. If $\pi(p)$ is a $U$-action-local proposition then $M, \pi \models K_V p \Rightarrow K_G p$.*

We remark that this result needs to be stated using group knowledge rather than distributed knowledge. Since $\models D_V \phi \Rightarrow K_V \phi$, we also have $M, \pi \models D_V p \Rightarrow K_G p$. However, it is not necessarily the case that $M, \pi \models D_V p \Rightarrow D_G p$. The reason has to do with the unsuitability of the notion of distributed knowledge for asynchronous settings: it is possible for $V$ to have observed the ordering of changes to observations at $G$, where, because of asynchrony, this ordering is not distributed knowledge to $G$. This is illustrated in the following example.

Consider the architecture $\mathcal{A}$ and system $M$ depicted in Figure 5. The architecture has four domains $A, B, C, D$. The system has states $s_0 \ldots s_3$, with initial state $s_0$. We suppose that there are actions $A = \{a, b\}$ with $\mathrm{dom}(a) = \mathrm{dom}(b) = A$. The effect of an action is only depicted when it changes the state, e.g., we draw the edge corresponding to $s_0 \cdot a = s_1$, but omit the self-loop corresponding to $s_1 \cdot a = s_1$. Associated with each state $s$ in the diagram is a tuple $(x, y)$, which is to be interpreted as giving the observation made by the agents $B$ and $C$ at that state. In particular, $\mathrm{obs}_B(s) = x$ and $\mathrm{obs}_C(s) = y$. The observations of the other agents $A$ and $D$ are given as follows: $\mathrm{obs}_A(s) = 0$ and $\mathrm{obs}_D(s) = (\mathrm{obs}_B(s), \mathrm{obs}_C(s))$ for all states $s$. It is not difficult to check that $M$ FTAO-complies with $\mathcal{A}$.

Suppose that we take $U = \{A\}$, $G = \{B, C\}$ and

$V = \{D\}$ and let $\pi(p) = \{a\}^+ b \{a, b\}^*$ be the proposition that there has been an occurrence of $a$ and an occurrence of $b$, but with the first $a$ preceding the first $b$. It is evident that $G$ is an observational barrier separating $U$ from $V$ and that $\pi(p)$ is $U$-action-local. We claim that $M, \pi, ab \models D_V p \wedge \neg D_G p$. For this, note that $\mathrm{view}_D(ab) = (0, 0)(1, 0)(1, 1)$. Thus, any sequence $\alpha$ such that $\mathrm{view}_D(ab) = \mathrm{view}_D(\alpha)$ must contain both an $a$ before any $b$ (to cause the observation $(1, 0)$) and a subsequent $b$ (to cause the observation $(1, 1)$), hence $M, \pi, \alpha \models p$. This means that $M, \pi, ab \models K_V p$, or equivalently, $M, \pi, ab \models D_V p$, On the other hand, we have $\mathrm{view}_B(ab) = 01 = \mathrm{view}_B(ba)$ and $\mathrm{view}_C(ab) = 01 = \mathrm{view}_C(ba)$, so $ab \approx_G^D ba$. Since $M, \pi, ba \not\models p$, this means that $M, \pi, ab \not\models D_G p$,

Intuitively, in this example, B detects the first occurrence of $a$ and C detects the first occurrence of $b$. Since D is able to synchronously observe everything that these agents observe, it knows the order of $a$ and $b$. However, this is not distributed knowledge to B and C because the semantics of distributed knowledge does not combine their information until after the run is complete, and the ordering information is, because of asynchrony, lost from the views of both agents. Similar issues with distributed knowledge in asynchronous settings are discussed have been noted previously [Moses and Bloom, 1994; van der Meyden, 2008].

## 9 Conclusion

We have introduced a new type of specification of the architecture of a multi-agent system, that enables upper bounds on causal effects and lower bounds on observed information to be expressed. We have given this type of specification a formal semantics that extends previous ideas from the literature on computer security. Using this specification format, we have shown by a number of examples that it is possible to derive security properties stated in epistemic logic from highly abstract assumptions that concern the architectural structure of a system and the behavior of certain *trusted components*, even when other components have unknown behavior. These results contribute to the formal foundations of the MILS vision that secure systems can be constructed by composition of trusted and untrusted components in the context of a specific architectural structure.

The term "agent architecture" is much used in work on intelligent agents and multi-agent systems; this area is surveyed in Wooldridge and Jennings [1994]. The emphasis in this literature is on systems built on some form of Artificial Intelligence capability. Many specific agent-oriented architectures have been proposed. Generally, these concern a much more concrete notion of ar-

chitecture than we have discussed here. For example, BDI-architectures [Rao and Georgeff, 1991] posit that an agent should be designed around the idea that it has beliefs, desires and intentions, and that the dynamics for these notions may be expressed in a multi-modal logic. Our work is considerably more abstract, and intended to apply broadly to all types of computational systems, and is therefore more closely related to the notion of architecture as studied in software engineering. We refer the reader to Chong and van der Meyden [2009] for an extended discussion of related work on architectural modelling formats from software engineering and on connections between architecture and security properties.

The literature on Bayesian nets [Pearl, 2000] uses diagrams that have some similarity to our diagrams restricted to the case of edges labelled $\top$. However, Bayesian nets are interpreted over a static set of states, rather than in a dynamic systems model such as we have considered. A more temporal view of causality is taken in some work in the literature, e.g., Halpern and Pearl [2001], but here the emphasis is on defining when one event can be understood as a cause of another, rather than on specifying the causal structure of a system.

Our objective in this paper has been to develop an extended architectural format that is able to express lower bounds on information flow and to demonstrate its usefulness for enforcing security properties expressed in epistemic logic. This contribution leaves open many questions for future research. It remains to develop practical proof techniques and implementation strategies that ensure that a system complies with our extended format. For example, it would be useful to generalize the refinement techniques developed in Chong and van der Meyden [2009] to the extended format. Further examples should also be developed to validate the MILS methodology to secure systems construction and our approach to its formalization.

## References

J. Alves-Foss, W. Harrison, P. Oman, and C. Taylor. The MILS architecture for high-assurance embedded systems. *International Journal of Embedded Systems*, 2 (3/4):239–47, Feb 2006.

C. Boettcher, R. DeLong, J. Rushby, and W. Sifre. The MILS component integration approach to secure information sharing. In *27th IEEE/AIAA Digital Avionics Systems Conference*, pages 1.C.2–1–1.C.2–14, Oct 2008.

S. Chong and R. van der Meyden. Using architecture to reason about information security. Submitted for publication, 2009.

K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In *Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge*, 1998.

J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach - Part II: Explanations. In *IJCAI*, pages 27–34, 2001.

Y. Moses and B. Bloom. Knowledge, timed precedence and clocks (preliminary report). In *Proc. ACM Symp. on Principles of Distributed Computing*, pages 294–303, 1994.

J. Pearl. *Causality: Models, reasoning and inference*. Cambridge University Press, 2000.

A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 473–484, 1991.

J. Rushby. Noninterference, transitivity and channel-control security policies. Technical report, SRI, 1992.

R. van der Meyden. On causality and distributed knowledge. In *KR 2008: Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, pages 209–212, 2008.

W. Vanfleet, R. Beckworth, B. Calloni, J. Luke, C. Taylor, and G. Uchenick. MILS:architecture for high assurance embedded computing. *Crosstalk: The Journal of Defence Engineering*, pages 12–16, Aug 2005.

M. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In *ECAI Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *LNCS*, pages 1–39. Springer, 1994.