Runtime Verification

Monitoring, Knowledge, and Uncertainty

MPRI Lecture Notes
(in progress)

Benedikt Bollig

February 3, 2026

Preface

These notes accompany an MPRI course given in the lecture series *Advanced Techniques of Verification*. They present runtime verification from the perspective of partial information, with a unifying treatment of monitoring, diagnosis, and opacity. Comments and corrections are welcome at `bollig@lmf.cnrs.fr`.

1

What Is Runtime Verification?

1.1 Context

Runtime verification studies how to observe a running system and raise an alarm when something goes wrong, or confirm that a behavior conforms with a specification. It can be applied even when a system model is not available. In this sense, it complements *model checking*: instead of exploring the full state space of a given model in advance, runtime verification analyzes executions (also called traces) as they are generated.

A monitor observes the execution step by step and produces a verdict based on the information available so far. If the observed behavior violates the specification, the monitor raises an alarm; if the behavior is guaranteed to satisfy the specification, it may confirm correctness.

A central theme of runtime verification is *uncertainty*. The system may be only partially known, or even unknown, and the monitor typically observes only a subset of the system's actions or propositions. Moreover, in reactive systems, executions are typically infinite, but at runtime only a finite prefix is available. As a result, verdicts may be inconclusive, reflecting the inherent limits of runtime observation.

Runtime verification shares this focus on uncertainty with closely related techniques such as diagnosis and opacity checking. The goal of these notes is to present a unifying treatment of these topics. To that end, we use epistemic logic as a common language to express what an observer knows (and does not know) from its observations.

In these notes, we actually use the term *runtime verification* in a broad sense, to refer to observation-based verification problems such as monitoring, diagnosis, and opacity, which all reason about what can be inferred from executions at runtime.

1.2 Model Checking, Diagnosis, Opacity, and Monitoring

We consider reactive systems that produce infinite executions. Throughout these notes, we assume that a (possibly abstract) model $\mathcal{S}$ of the system is available. Depending on the amount of information encoded in this model, we distinguish three idealized cases.

- *White-box systems*, where the model accurately reflects the implementation.
- *Gray-box systems*, where the model intentionally abstracts or hides information, for instance by introducing nondeterminism to represent unknown environment behavior, partial observability, or underspecification.
- *Black-box systems*, where no reliable model is available and the system can only be observed through its executions.

Model checking. When a sufficiently precise model of the system is available (white box, or controlled gray box), correctness questions can be addressed using *model checking*. Given a temporal specification φ, the model-checking problem asks whether

$$\mathcal{S} \models \varphi,$$

that is, whether φ holds on *all* infinite executions of $\mathcal{S}$.

In gray-box models, however, nondeterminism often represents uncertainty rather than real implementation choices. In this case, universal quantification over all executions may be too strong or unfounded. A model may include executions that are possible in the abstraction but impossible in reality, or behaviors that will only be resolved at runtime. As a result, a purely offline verdict may be misleading.

From offline verification to runtime observation. An alternative is to observe executions as they occur. At runtime, the system produces a concrete execution, but an observer typically sees only a projection. Based on the observed prefix, the observer reasons about the set of executions of $\mathcal{S}$ that are compatible with what has been seen so far.

This naturally leads to epistemic reasoning. At any time, the observer may or may not *know* that a given property holds, depending on whether it holds on all executions consistent with the current observation.

Diagnosis. In diagnosis, the goal is not to decide an arbitrary specification, but to detect the occurrence of a particular event, such as a fault. Let φ be a property representing such an event.

A diagnoser should eventually know that the event occurred, whenever it did. The existence of a diagnoser is naturally expressed by the property

$$\Phi_{\mathsf{diagnosable}}(\varphi) = \mathsf{G}\big(\varphi \to \mathsf{FK}_a\mathsf{P}\varphi\big),$$

which can be read as

> "It is globally true (G) that if φ holds, then eventually agent a knows that φ held in the past."

Thus, the existence of a diagnoser is a model-checking question, while the diagnoser itself operates online on the observation stream.

In diagnosis, fault propositions are typically *unobservable* to the observer. This does not mean that faults are absent from the system model. Rather, faults represent internal events or mode changes that may influence the future behavior of the system, even if they are not directly visible. Modeling faults explicitly allows us to define correctness of diagnosis and to reason about whether their occurrence can be inferred indirectly from observations.

Example 1.1 (Diagnosable vs. non-diagnosable systems)

Let $AP = \{p, r, e\}$ be a set of propositions and $\Sigma = 2^{AP}$. A letter $\sigma \in \Sigma$ specifies which propositions hold at a discrete time step. It is also referred to as an *event*. Think of e as "an error occurred". An execution (or trace) is an infinite sequence in Σ^ω (i.e., it is an infinite word over Σ), and the behavior of a system $\mathcal{S}$ is given by its language $L(\mathcal{S}) \subseteq \Sigma^\omega$.

Given an event with letter $\sigma \in \Sigma$, agent a observes only $\sigma \cap AP_a$, where $AP_a = \{p, r\}$, i.e., the error proposition e is unobservable.

Consider the three systems $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ with the following languages:

$$
\begin{aligned}
L(\mathcal{S}_1) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}\{r\}^\omega \\
L(\mathcal{S}_2) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}^\omega \\
L(\mathcal{S}_3) &= \{p\}^\omega + \{p\}^*\{e\}\{p\}^\omega \\
L(\mathcal{S}_4) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}^*\{r\}^\omega
\end{aligned}
$$

where we use $\alpha\beta^\omega$ as shorthand for the corresponding ω-language.

For example, the infinite word $\{p\}\{p\}\{p,e\}\{p\}\{r\}\{r\}\{r\}\ldots$ is a possible trace of $\mathcal{S}_1$. Agent a observes the projection $\{p\}\{p\}\{p\}\{p\}\{r\}\{r\}\{r\}\ldots$. Since observing r eventually allows a to infer that an error has occurred, $\mathcal{S}_1$ is diagnosable for the property $\varphi = e$.

In contrast, $\mathcal{S}_2$ is not diagnosable: faulty and non-faulty executions are observationally indistinguishable, as a observes only sequences of $\{p\}$ in both cases.

The system $\mathcal{S}_3$ is again diagnosable. Although the error proposition e is unobservable and all observations after the error are $\{p\}$, the error occurs at a position where the observer sees $\emptyset$, whereas non-faulty executions produce $\{p\}$ from the first step. Under the synchronous perfect-recall semantics, the observer remembers the number of steps, so these prefixes are distinguishable. Hence agent a can infer that an error has occurred, and $\mathcal{S}_3$ is diagnosable.

The system $\mathcal{S}_4$ is also diagnosable. Indeed, on every faulty execution, the symbol $\{r\}$ eventually occurs (and then persists forever), whereas on non-faulty executions only $\{p\}$ is observed. Thus, once $\{r\}$ is observed, agent a can conclude that an error has occurred. However, $\mathcal{S}_4$ illustrates that *unbounded* diagnosis does not imply a *uniform* delay bound: after an error, the system may produce $\{p\}$ for an arbitrarily long finite time before switching to $\{r\}^\omega$. Therefore, for every $D \in \mathbb{N}$ there exists a faulty execution in which agent a cannot yet infer the error within D further steps.

We conclude that

$$\mathcal{S}_1 \models \Phi_{\mathsf{diagnosable}}(e) \quad \text{and} \quad \mathcal{S}_3 \models \Phi_{\mathsf{diagnosable}}(e) \quad \text{and} \quad \mathcal{S}_4 \models \Phi_{\mathsf{diagnosable}}(e),$$

whereas

$$\mathcal{S}_2 \not\models \Phi_{\mathsf{diagnosable}}(e).$$

Note that only $\mathcal{S}_1$ and $\mathcal{S}_3$ satisfy a bounded form of diagnosability: $\mathsf{G}(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{P} e)$ for suitable fixed D, where X^D means *after D steps*.

Opacity. Opacity addresses the dual problem of diagnosis. Here, the secret is not a current-state fact, but the *occurrence of an event* at some point in the execution. The goal of opacity is to ensure that the observer can never infer that this event has occurred.

Accordingly, we define opacity with respect to a past property. Given a secret event φ, an epistemic opacity requirement is

$$\Phi_{\mathsf{opaque}}(\varphi) = \mathsf{G}\,\neg \mathsf{K}_a \mathsf{P} \varphi.$$

Thus, opacity requires that at all times the observer does not know that the secret event has occurred in the past. Note that this does not prevent the observer from sometimes knowing that the secret event has not occurred.

As before, this is a model-checking problem on the system $\mathcal{S}$.

Example 1.2 (Past opacity)

Let $AP = \{p, s\}$ and $\Sigma = 2^{AP}$. Agent a observes only p, i.e., $AP_a = \{p\}$. The proposition s represents a secret internal event and is unobservable.

Consider the system $\mathcal{S}$ with language

$$L(\mathcal{S}) = \{p\}^\omega + \{p\}^*\{p, s\}\{p\}^\omega.$$

Let the secret event be $\varphi = s$, and consider the past secret $\mathsf{P}s$ ("s has occurred at

some point in the past"). Then $\mathcal{S}$ is opaque with respect to φ and agent a, since for every observation prefix consisting only of $\{p\}$, there exists an indistinguishable execution in which s has not occurred. Hence,

$$\mathcal{S} \models \Phi_{\mathsf{opaque}}(\varphi).$$

Monitoring. A monitor observes a finite prefix of an execution and produces a verdict based on the information available so far. In the presence of partial observability, such verdicts naturally correspond to what the observer can infer from the observations. Accordingly, we model monitoring using three-valued verdicts derived from epistemic reasoning.

Let φ be a temporal property interpreted at the initial position of the execution, and let $\hat{\varphi} = \mathsf{P}(\mathit{first} \wedge \varphi)$. Given an observation prefix, the monitor produces:

- the verdict true if $\mathsf{K}_a\hat{\varphi}$ holds, i.e., the observer knows that φ holds at the initial position;
- the verdict false if $\mathsf{K}_a\neg\hat{\varphi}$ holds, i.e., the observer knows that φ does not hold;
- the verdict ? otherwise, reflecting remaining uncertainty.

Monitoring thus tracks whether the observer has obtained sufficient knowledge to decide the truth of φ.

Example 1.3 (A simple epistemic monitor)

Let $AP = \{p, r, e\}$ and $\Sigma = 2^{AP}$. As before, think of e as "an error occurred". We consider a single agent a that can observe all propositions, i.e., $AP_a = AP$.

Consider the system $\mathcal{S}$ with

$$L(\mathcal{S}) = \{p\}^*\{r\}\{p\}^\omega + \{p\}^*\{e\}\{p\}^\omega.$$

That is, after some (possibly empty) prefix of $\{p\}$, exactly one of $\{r\}$ or $\{e\}$ occurs, and then only $\{p\}$ forever.

Let the property be $\varphi = \neg\mathsf{F}e$, expressing that no error ever occurs. As long as the observer sees only $\{p\}$, both executions in which an error will occur and executions in which no error occurs remain compatible with the observations. Hence, neither $\mathsf{K}_a\hat{\varphi}$ nor $\mathsf{K}_a\neg\hat{\varphi}$ holds, and the monitor outputs ?.

Upon observing the first non-$\{p\}$ letter, epistemic uncertainty is resolved. If the letter is $\{e\}$, then the property is violated, $\mathsf{K}_a\neg\hat{\varphi}$ holds, and the monitor outputs

false. If the letter is $\{r\}$, then it is guaranteed that no error will ever occur, $\mathsf{K}_a\hat{\varphi}$ holds, and the monitor outputs true.

A unifying perspective. Model checking, runtime verification, diagnosis, and opacity can all be analyzed within a common epistemic temporal framework over the same system model. The key difference lies in *when* and *how* the result is used.

Model checking provides an offline verdict about all executions. Diagnosis and opacity are typically formulated as epistemic temporal properties whose satisfaction can be decided offline. Monitoring, in contrast, produces verdicts online from finite observations, but relies on similar epistemic reasoning about what an observer can infer.

In this sense, runtime verification does not replace offline analysis. Instead, offline analysis is used to construct monitors, diagnosers, or observers, which are then applied at runtime to the observed execution.

1.3 Take-Away

- Runtime verification reasons about system correctness from *observed executions*, rather than from complete system models.
- Partial observation and infinite executions introduce uncertainty, which can be captured naturally using epistemic notions of knowledge.
- Model checking, diagnosis, opacity, and monitoring can all be analyzed using epistemic temporal reasoning over the same abstract system model.
- Diagnosis and opacity express availability or impossibility of knowledge, while monitoring general properties raises questions about when definitive verdicts can be reached.
- Although monitors operate online, their existence and correctness are typically established by an *offline analysis* of the system model.

1.4 Further Reading

Runtime verification has been extensively studied as a lightweight alternative to exhaustive model checking, in particular for linear-time temporal logics [2, 10]. For background on model checking and temporal logics, we refer to the standard textbook [1]. Diagnosis and opacity originate from the discrete-event systems community, where they are traditionally formulated in terms of observation-based distinguishability of behaviors [9, 11]. Opacity has also been studied on transition systems and from an information-flow perspective [4].

Epistemic logic provides a unifying formalism for reasoning about such distinguishability, by making explicit what an observer can infer from partial observations. The interaction between temporal and epistemic operators has been studied in the context of knowledge and time under perfect-recall semantics (e.g., [7]). In these notes, we make use of this connection and adopt epistemic temporal logic as a common language to relate monitoring, diagnosis, and opacity in a runtime setting.

2

Observation, Knowledge, and Uncertainty

2.1 Words, Languages, and Moore Machines

We let $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ and $\mathbb{N}_{\geq 1} = \{1, 2, 3, \ldots\}$.

A possible behavior of a reactive system is naturally viewed as a sequence of observations over time. Such behaviors are represented as *words*, also called *traces*, which may be finite or infinite sequences over a given alphabet.

Definition 2.1 (Words)

Let Σ be a finite alphabet. A *finite word* over Σ is a sequence $u = \sigma_1 \ldots \sigma_n$ with $n \in \mathbb{N}$ and $\sigma_i \in \Sigma$ for $i \in \{1, \ldots, n\}$. We let $|u|$ refer to the *length* n of u. If $n = 0$, then u is the empty word, denoted by ε. By Σ^*, we denote the set of all finite words over Σ. The set of nonempty finite words is denoted by $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.

An *infinite word* over Σ is a sequence $w = \sigma_1\sigma_2\sigma_3 \ldots$ with $\sigma_i \in \Sigma$ for $i \in \mathbb{N}_{\geq 1}$. The set of infinite words is denoted by Σ^ω. Moreover, we let $\Sigma^\infty = \Sigma^* \cup \Sigma^\infty$.

Given $w = \sigma_1\sigma_2\sigma_3 \ldots \in \Sigma^\omega$ and $k \in \mathbb{N}$, we let $w_{\leq k}$ refer to the finite prefix $\sigma_1 \ldots \sigma_k \in \Sigma^*$ of length k. Note that $w_{\leq 0} = \varepsilon$. For a finite word $u = \sigma_1 \ldots \sigma_n \in \Sigma^*$ and $k \in \{0, \ldots, n\}$, the prefix of length k of u is defined accordingly, as $u_{\leq k} = \sigma_1 \ldots \sigma_k$.

A *language* is a set of words. We mainly consider languages $L \subseteq \Sigma^*$ over finite words, and languages $L \subseteq \Sigma^\omega$ over infinite words. For $L \subseteq \Sigma^\infty$, we let $\mathit{Pref}^+(L)$ refer to $\{w_{\leq k} \mid w \in L \text{ and } k \in \mathbb{N}_{\geq 1}\}$, i.e., the set of *nonempty* finite prefixes of words in L.

Example 2.2 (A simple word)

Let $AP = \{p, r\}$, i.e., $\Sigma = \{\emptyset, \{p\}, \{r\}, \{p, r\}\}$. Then, $\{p\}\emptyset\{p\}\emptyset\{p\}\emptyset \ldots \in \Sigma^\omega$ is an infinite word. Moreover, $\{p\}\{p, r\}\emptyset \in \Sigma^*$ is a finite word of length 3.

While we will later define Büchi automata as a central automata model running over infinite words, we already introduce deterministic automata with output, also known as Moore machines. These will play a crucial role as runtime monitors and as building blocks in constructions over Büchi automata.

Definition 2.3 (Moore machine)

A *Moore machine* over an input alphabet Σ and an output alphabet Γ is a tuple

$$\mathcal{A} = (Q, \iota, \delta, o)$$

where

- Q is a finite set of *states*,
- $\iota \in Q$ is the *initial state*,
- $\delta \colon Q \times \Sigma \to Q$ is a *transition function*,
- $o \colon Q \to \Gamma$ is an *output mapping*.

We extend δ inductively to $\delta^* \colon Q \times \Sigma^* \to Q$ by $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$ for all $q \in Q$, $w \in \Sigma^*$, and $\sigma \in \Sigma$. As its semantics, the Moore machine $\mathcal{A}$ defines the following mapping:

$$[\![\mathcal{A}]\!] \colon \begin{cases} \Sigma^* \to \Gamma \\ u \mapsto o(\delta^*(\iota, u)) \end{cases}$$

In other words, it scans the input word and ouputs the symbol associated with the resulting state.

Special case: deterministic finite automata. For $\Gamma = \{0, 1\}$, a Moore machine $\mathcal{A} = (Q, \iota, \delta, o)$ can be viewed as a deterministic finite automaton over Σ by identifying the set of accepting states as

$$F = \{\, q \in Q \mid o(q) = 1 \,\}.$$

In this case, we will equivalently write $\mathcal{A} = (Q, \iota, \delta, F)$ and understand it as a recognizer for the language

$$L(\mathcal{A}) = \{\, u \in \Sigma^* \mid \delta^*(\iota, u) \in F \,\}.$$

2.2 Propositions, Agents, and Partial Observability

In our setting, the behavior of a reactive system is modeled as an infinite word over an alphabet whose letters indicate which atomic propositions hold at each time step. A single position of such a word, together with its associated set of propositions, is called an *event*.

In many systems, an observer cannot see all aspects of an event. Instead, each agent has access only to a subset of the atomic propositions. As a consequence, different system executions may be indistinguishable from the point of view of a given agent. This form of partial observability is formalized by projecting global executions onto the propositions visible to the agent and will later form the basis of our epistemic semantics.

Throughout these notes, unless stated otherwise, we fix a finite set of atomic propositions AP and let $\Sigma = 2^{AP}$ be the corresponding global alphabet. We also fix a finite set Ag of *agents*. For each agent $a \in Ag$, let $AP_a \subseteq AP$ denote the set of propositions observable by a, and let $\Sigma_a = 2^{AP_a}$ be the corresponding observation alphabet.

Definition 2.4 (Observable propositions)

The observation function for agent a is

$$obs_a : \begin{cases} \Sigma \to \Sigma_a \\ \sigma \mapsto \sigma \cap AP_a \end{cases}$$

It is extended pointwise to words in Σ^* and Σ^ω. Note that obs_a is then length-preserving and that $obs_a(\varepsilon) = \varepsilon$.

The observation function formalizes the local view of an agent at a single step. Given a global event, the agent retains only the information carried by the propositions it can observe and ignores the rest. Extending observations from single events to words allows us to compare entire executions from the agent's perspective.

Example 2.5 (Observation of words)

Let $AP = \{p, r, e\}$ and let agent a observe only $AP_a = \{p, r\}$.

Consider the finite word

$$u = \{p\}\{p, e\}\{r\} \in \Sigma^*.$$

Then

$$obs_a(u) = \{p\}\{p\}\{r\}.$$

If an event contains only unobservable propositions, the observation is empty. For example, for

$$u' = \{p\}\{e\}\{r\} \in \Sigma^*,$$

we have

$$obs_a(u') = \{p\}\emptyset\{r\}.$$

Using observations, we can now formalize when an agent is unable to distinguish between two executions. Intuitively, two executions are indistinguishable for an agent if they give rise to exactly the same sequence of observations.

Definition 2.6 (Indistinguishability for finite words)

Two finite words $u, u' \in \Sigma^*$ are *indistinguishable to agent* a, written $u \sim_a u'$, if they have the same observation for agent a:

$$u \sim_a u' \quad \text{if} \quad obs_a(u) = obs_a(u')$$

In particular, indistinguishable words are of the same length.

Note that indistinguishability is an equivalence relation on finite words.

Example 2.7 (Indistinguishable finite words)

Let $AP = \{p, r, e\}$ and let agent a observe $AP_a = \{p, r\}$, as in Example 1.1. Consider the finite words

$$u = \{p\}\{p, e\}\{p\} \qquad \text{and} \qquad u' = \{p\}\{p\}\{p\}.$$

Since the proposition e is unobservable for agent a, we have

$$obs_a(u) = obs_a(u') = \{p\}\{p\}\{p\}.$$

Hence, $u \sim_a u'$.

When reasoning about properties at a specific time point, it is not sufficient to compare complete executions. We therefore lift indistinguishability to pairs consisting of an execution and a position. Two such pairs are indistinguishable if the agent has seen the same observations up to that position.

Definition 2.8 (Indistinguishability for pointed infinite words)

A *pointed word* is a pair (w, k) where $w \in \Sigma^\omega$ and k is a position in w, i.e., $k \in \mathbb{N}_{\geq 1}$.

Two pointed words (w, k) and (w', k) are *indistinguishable to agent* a (perfect recall), written $(w, k) \sim_a (w', k)$, if their prefixes up to position k are indistinguishable:

$$(w, k) \sim_a (w', k) \quad \text{if} \quad w_{\leq k} \sim_a w'_{\leq k}$$

Note (Synchronous vs. asynchronous observation)

In these notes, we consider a *synchronous* observation semantics: system progress is observable by all agents. Formally, at each step, every agent receives an observation, possibly empty. Equivalently, one may assume a distinguished atomic proposition *tick* that holds at every step and is observable by all agents. Thus, while a transition may be uninformative, it is never (epistemically) silent.

Under this assumption, indistinguishability is naturally defined between prefixes of equal length, as in Definition 2.6.

An alternative *asynchronous* observation semantics can be obtained by letting agents observe only steps carrying nonempty observations. Formally, one replaces $obs_a \colon \Sigma^* \to \Sigma_a^*$ by the projection that erases events σ with $obs_a(\sigma) = \emptyset$, and accordingly extend this to indistinguishability by

$$(w, k) \sim_a (w', k') \quad \iff \quad obs_a(w_{\leq k}) = obs_a(w'_{\leq k'}) \,.$$

In this case, agents may be uncertain about the number of system steps or the elapsed time.

All epistemic definitions and constructions presented in these notes extend to this asynchronous variant by replacing the observation function accordingly. For simplicity, we focus on the synchronous case.

2.3 A Base System Model

Systems are characterized by their set of infinite traces, i.e., the infinite words they possibly generate. These can, for example, be described in terms of ω-regular expressions (cf. Chapter 1) or in terms of automata. For an algorithmic treatment and also from a modeling point of view, it is convenient to use Büchi automata.

Definition 2.9 (Büchi automaton)

A *Büchi automaton* over Σ is a tuple

$$\mathcal{A} = (Q, \iota, \Delta, F)$$

where

- Q is a finite set of *states* and $\iota \in Q$ is an *initial state*,

- $\Delta \subseteq Q \times \Sigma \times Q$ is a *transition relation*,
- $F \subseteq Q$ is a set of *accepting states*.

Let $q, q' \in Q$ be states. For $\sigma \in \Sigma$, we write $q \xrightarrow{\sigma} q'$ if $(q, \sigma, q') \in \Delta$. A *run* of $\mathcal{A}$ is an infinite sequence of the form $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} \dots$ such that $q_0 = \iota$.

A *run from* q (or q run) is defined analogously, except that $q_0 = q$. However, unless stated otherwise, the term *run* always refers to a run starting in the initial state.

We let $\pi(i)$ refer to q_i for all $i \in \mathbb{N}$. Run π is called *accepting* if $q_i \in F$ for infinitely many $i \in \mathbb{N}$. The *induced trace* is $\sigma_1\sigma_2\sigma_3 \dots \in \Sigma^\omega$, denoted by $\mathit{trace}(\pi)$. We also say that π is a run *over* w.

The *language* of $\mathcal{A}$ is defined as

$$L(\mathcal{A}) = \{\, \mathit{trace}(\pi) \mid \pi \text{ is an accepting run of } \mathcal{A} \,\}.$$

It is convenient to also introduce finite runs. A *finite run* of $\mathcal{A}$ is a finite sequence of the form $\pi = q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \dots \xrightarrow{\sigma_n} q_n$, where $n \in \mathbb{N}$, such that $q_0 = \iota$. We also set $\pi(i) = q_i$ for all $i \in \{0, \dots, n\}$. Moreover, we let $\mathit{trace}(\pi) = \sigma_1 \dots \sigma_n$, and we say that π is a run over $\mathit{trace}(\pi)$. We do not define acceptance for finite runs.

Note

For technical convenience in some constructions, we also use *generalized Büchi automata* (GBAs), where the acceptance condition is given by a set of accepting sets $\mathcal{F} = \{F_1, \dots, F_m\}$ with $F_i \subseteq Q$. A run is accepting if, for each $i \in \{1, \dots, m\}$, it visits some state from F_i infinitely often. Every GBA can be converted to a standard Büchi automaton with at most $m \cdot |Q|$ states (cf. [1]).

We define

$$\mathit{Obs}_a(\mathcal{A}) = \{\, \mathit{obs}_a(u) \mid u \in \mathit{Pref}^+(L(\mathcal{A})) \,\}$$

for the set of all (nonempty) observation sequences of agent a that are compatible with some finite execution of $\mathcal{A}$.

Note

System states are not observed directly. Instead, the observer sees only the projected execution trace over the observation alphabet Σ_a. Classical runtime monitoring corresponds to a maximally permissive special case, where the system model consists of a single (accepting) state with a self-loop for every event $\sigma \in \Sigma$.

Example 2.10 (Four trace languages under partial observation)

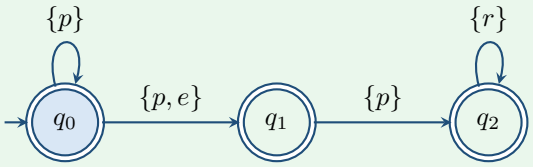
Let us revisit Example 1.1 and provide Büchi automata for the languages considered there.

Recall that $AP = \{p, r, e\}$ and $\Sigma = 2^{AP}$. A letter $\sigma \in \Sigma$ specifies which propositions hold at a discrete time Fix an agent a with $AP_a = \{p, r\}$, so the proposition e is unobservable.

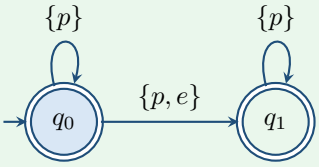
We consider four systems $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ given by the following languages:

$$
\begin{aligned}
L(\mathcal{S}_1) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}\{r\}^\omega \\
L(\mathcal{S}_2) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}^\omega \\
L(\mathcal{S}_3) &= \{p\}^\omega + \{p\}^*\{e\}\{p\}^\omega \\
L(\mathcal{S}_4) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}^*\{r\}^\omega
\end{aligned}
$$

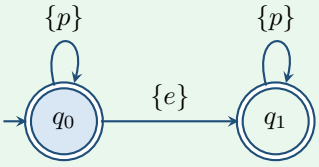
Each language is generated by the following (small) Büchi automaton (the initial state has an incoming arrow and final states are depicted as a double circle):



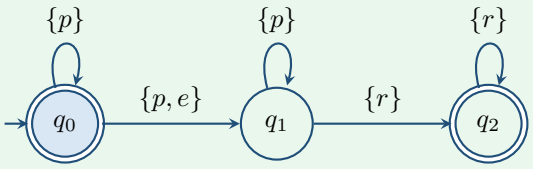
(a) Automaton for $\mathcal{S}_1$



(b) Automaton for $\mathcal{S}_2$



(c) Automaton for $\mathcal{S}_3$



(d) Automaton for $\mathcal{S}_4$

2.4 Epistemic Linear-Time Temporal Logic

We now formalize the epistemic temporal logic outlined in the introduction. The logic allows us to specify temporal properties of systems under partial observability, and to reason explicitly about what agents know based on their observations. In particular, it provides a logical framework for formalizing diagnosability or opacity properties.

Definition 2.11 (Epistemic LTL)

Formulas from *Epistemic Linear-time Temporal Logic*, denoted by LTL$^{\sim}$, are given by the following grammar:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathsf{U}^{+} \varphi \mid \varphi \mathsf{S}^{+} \varphi \mid \mathsf{K}_a \varphi$$

where $p \in AP$ is a proposition and $a \in Ag$ is an agent.

Formulas are interpreted over pointed words. Recall that a pointed word consists of an execution together with a distinguished reference position (cf. Definition 2.8). This position separates the finite prefix that has already been observed from the infinite future that is yet to unfold. Both temporal and epistemic operators are evaluated relative to this reference position.

Definition 2.12 (Satisfaction)

Let $\mathcal{S}$ be a Büchi automaton over Σ (representing a system). Let $w = \sigma_1\sigma_2\sigma_3 \ldots \in L(\mathcal{S})$ and $k \in \mathbb{N}_{\geq 1}$. Satisfaction $\mathcal{S}, w, k \models \varphi$ for a formula φ is inductively defined as follows:

- $\mathcal{S}, w, k \models p$ if $p \in \sigma_k$
- $\mathcal{S}, w, k \models \neg\varphi$ if $\mathcal{S}, w, k \not\models \varphi$
- $\mathcal{S}, w, k \models \varphi \wedge \psi$ if $\mathcal{S}, w, k \models \varphi$ and $\mathcal{S}, w, k \models \psi$

- $\mathcal{S}, w, k \models \varphi \mathsf{U}^+ \psi$ if there exists j with $k < j$ such that
  - $\mathcal{S}, w, j \models \psi$ and
  - for all i with $k < i < j$, we have $\mathcal{S}, w, i \models \varphi$
- $\mathcal{S}, w, k \models \varphi \mathsf{S}^+ \psi$ if there exists j with $1 \leq j < k$ such that
  - $\mathcal{S}, w, j \models \psi$ and
  - for all i with $j < i < k$, we have $\mathcal{S}, w, i \models \varphi$
- $\mathcal{S}, w, k \models \mathsf{K}_a \varphi$ if for all $w' \in L(\mathcal{S})$ with $(w, k) \sim_a (w', k)$, we have $\mathcal{S}, w', k \models \varphi$

Moreover, we write $\mathcal{S} \models \varphi$ if, for every $w \in L(\mathcal{S})$, we have $\mathcal{S}, w, 1 \models \varphi$.

Note (Strict temporal operators)

We take the *strict* temporal operators U^+ and S^+ as primitives. Thus, $\varphi \mathsf{U}^+ \psi$ requires ψ to hold at a strictly future position, and $\varphi \mathsf{S}^+ \psi$ at a strictly past one. The left-hand side φ is required to hold only at positions strictly between the current position and the witnessing position. Its truth at the current position is irrelevant.

This avoids separate next and previous operators as primitives, since they can be defined using strict until and since. More importantly, it anticipates the timed setting. In timed logics, strictness is needed to constrain the elapsed time to the closest future or past occurrence of a subformula. Using strict operators already in the untimed logic yields a uniform semantics and simplifies the timed extension.

Derived temporal operators. Using the strict operators U^+ and S^+, we introduce the following abbreviations, which include the standard non-strict operators:

$$
\begin{array}{rcl@{\qquad}rcl}
\top & := & p \vee \neg p & \bot & := & \neg \top \\
\varphi \vee \psi & := & \neg(\neg\varphi \wedge \neg\psi) & \varphi \rightarrow \psi & := & \neg\varphi \vee \psi \\
\mathsf{X}\varphi & := & \bot \mathsf{U}^+ \varphi & \mathsf{Y}\varphi & := & \bot \mathsf{S}^+ \varphi \\
\varphi \mathsf{U} \psi & := & \psi \vee (\varphi \wedge (\varphi \mathsf{U}^+ \psi)) & \varphi \mathsf{S} \psi & := & \psi \vee (\varphi \wedge (\varphi \mathsf{S}^+ \psi)) \\
\mathsf{F}\varphi & := & \top \mathsf{U} \varphi & \mathsf{P}\varphi & := & \top \mathsf{S} \varphi \\
\mathsf{G}\varphi & := & \neg \mathsf{F} \neg \varphi & \mathsf{H}\varphi & := & \neg \mathsf{P} \neg \varphi \\
\mathit{first} & := & \neg \mathsf{Y} \top & \mathsf{W}_a \varphi & := & \mathsf{K}_b \varphi \vee \mathsf{K}_b \neg \varphi
\end{array}
$$

Here, a is an agent and $p \in AP$ any arbitrary proposition. In particular, the formula $\mathsf{W}_a \varphi$ denotes that a knows *whether* φ holds.

In particular, $\mathsf{X}\varphi$ requires that φ holds at the immediately next position, while $\mathsf{Y}\varphi$ requires that an immediately preceding position exists and that φ holds at that position.

Bounded temporal operators. For runtime verification and diagnosability, we also introduce bounded eventually and past operators. For $D \in \mathbb{N}$:

$$\begin{aligned} \mathsf{F}^{\leq D}\varphi &:= \varphi \vee \mathsf{X}\varphi \vee \mathsf{X}^2\varphi \vee \cdots \vee \mathsf{X}^D\varphi && \text{(eventually within } n \text{ steps)} \\ \mathsf{P}^{\leq D}\varphi &:= \varphi \vee \mathsf{Y}\varphi \vee \mathsf{Y}^2\varphi \vee \cdots \vee \mathsf{Y}^D\varphi && \text{(sometime in last } D \text{ steps)} \end{aligned}$$

where $\mathsf{X}^0\varphi = \varphi$ and $\mathsf{X}^{i+1}\varphi = \mathsf{X}(\mathsf{X}^i\varphi)$, and similarly for Y^i.

Semantically, $\mathcal{S}, w, k \models \mathsf{F}^{\leq n}\varphi$ holds iff there exists j with $k \leq j \leq k + n$ such that $\mathcal{S}, w, j \models \varphi$.

Bounded always operators $\mathsf{G}^{\leq n}\varphi := \neg\mathsf{F}^{\leq n}\neg\varphi$ and $\mathsf{H}^{\leq n}\varphi := \neg\mathsf{P}^{\leq n}\neg\varphi$ can be defined analogously.

Definition 2.13 (Logical equivalence)

LTL$^\sim$ formulas φ and ψ are *logically equivalent*, written

$$\varphi \equiv \psi,$$

if for every Büchi automaton $\mathcal{S}$, every word $w = \sigma_1\sigma_2\sigma_3\ldots \in L(\mathcal{S})$, and every position $k \in \mathbb{N}_{\geq 1}$,

$$\mathcal{S}, w, k \models \varphi \quad \Longleftrightarrow \quad \mathcal{S}, w, k \models \psi.$$

Note

Throughout these notes, we fix a system $\mathcal{S}$ and restrict attention to executions of $\mathcal{S}$. In particular, whenever satisfaction $\mathcal{S}, w, k \models \varphi$ is evaluated, we assume $w \in L(\mathcal{S})$. Thus, $L(\mathcal{S})$ plays the role of the universe of possible executions. We will restate this convention explicitly whenever it is relevant.

Exercise 2.14 (Basic axioms)

Let $\mathcal{S}$ be a system, let $a \in Ag$, and let φ be a formula.

(1) Show that

$$\mathcal{S} \models \varphi \quad \Longleftrightarrow \quad \mathcal{S} \models \mathsf{K}_a\varphi.$$

(2) Show that, for all $w \in L(\mathcal{S})$ and all $k \in \mathbb{N}_{\geq 1}$, we have

(a) $\mathcal{S}, w, k \models \mathsf{K}_a\varphi \implies \mathcal{S}, w, k \models \varphi$ (Veridicality)

(b) $\mathcal{S}, w, k \models \mathsf{K}_a\varphi \implies \mathcal{S}, w, k \models \mathsf{K}_a\mathsf{K}_a\varphi$ (Positive introspection)

(c) $\mathcal{S}, w, k \models \neg\mathsf{K}_a\varphi \implies \mathcal{S}, w, k \models \mathsf{K}_a\neg\mathsf{K}_a\varphi$ (Negative introspection)

(d) $\mathcal{S}, w, k \models \mathsf{K}_a\varphi \implies \mathcal{S}, w, k \models \mathsf{GK}_a\mathsf{P}\varphi$ (Non-forgetting)

(3) Show that the converse of the implication in (2a) does not hold in general.

Example 2.15 (Diagnosability properties)

We consider diagnosability properties for the systems $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ from Example 2.10, where e is unobservable to agent a.

Immediate diagnosability. $\mathsf{G}(e \to \mathsf{K}_a e)$ requires that errors are immediately detected. This fails for all four systems, as the letter $\{p, e\}$ is observed as $\{p\}$.

Eventual diagnosability. $\mathsf{G}(e \to \mathsf{FK}_a\mathsf{P}e)$ requires that errors are eventually detected. Systems $\mathcal{S}_1$, $\mathcal{S}_3$, and $\mathcal{S}_4$ satisfy this property: in $\mathcal{S}_1$, the observation $\{r\}$ occurs only on faulty executions; in $\mathcal{S}_3$, the error occurs when the observer sees $\emptyset$, distinguishing it from non-faulty executions; in $\mathcal{S}_4$, faulty executions eventually produce $\{r\}^\omega$ while non-faulty ones remain in $\{p\}^\omega$. System $\mathcal{S}_2$ fails this property, as faulty and non-faulty executions remain indistinguishable.

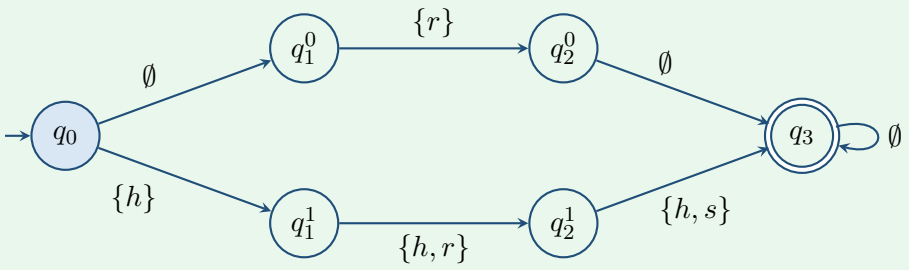
Bounded diagnosability. For fixed $D \in \mathbb{N}$, the formula $\mathsf{G}(e \to \mathsf{X}^D\mathsf{K}_a\mathsf{P}e)$ requires detection within D steps. Systems $\mathcal{S}_1$ and $\mathcal{S}_3$ satisfy this for suitable D. System $\mathcal{S}_4$ shows that eventual diagnosability does not imply bounded diagnosability: after an error, the system may produce $\{p\}$ for arbitrarily many steps before switching to $\{r\}^\omega$.

Example 2.16 (Nested knowledge with two agents)

We present a system where agent a knows that agent b knows a fact, although agent a does not know the fact itself.

Let $AP = \{h, r, s\}$ and $Ag = \{a, b\}$ with $AP_a = \{r\}$ and $AP_b = \{s\}$. Thus h is unobservable to both agents.

The system $\mathcal{S}$ nondeterministically chooses $h \in \{0, 1\}$, emits r, then emits s iff $h = 1$, and remains silent thereafter. Formally, $\mathcal{S}$ is the Büchi automaton below:



The language $L(\mathcal{S})$ contains two words, namely are $\emptyset\{r\}\emptyset\emptyset\ldots$ (when $h = 0$) and $\{h\}\{h, r\}\{h, s\}\emptyset\emptyset\ldots$ (when $h = 1$). Agent a observes only $\{r\}$ at position 2 in both cases, while agent b observes $\{s\}$ at position 3 iff $h = 1$.

Recall that $\mathsf{W}_b h := \mathsf{K}_b h \vee \mathsf{K}_b \neg h$. Consider

$$\varphi = \mathsf{XX}\big(\mathsf{K}_a \mathsf{W}_b h \wedge \neg \mathsf{W}_a h\big).$$

At position 3, agent b's observation determines h, so $\mathsf{W}_b h$ holds. Agent a has seen only $\{r\}$ (identical in both runs), so $\neg \mathsf{W}_a h$ holds. However, agent a knows the system structure and knows that agent b's observation suffices to determine h, hence $\mathsf{K}_a \mathsf{W}_b(h)$ holds. Thus $\mathcal{S} \models \varphi$.

So far, formulas have been interpreted at a concrete execution and a reference position. From the perspective of an agent, however, the actual execution is not directly accessible. Instead, the agent only has access to the finite observation it has made so far.

We therefore provide a satisfaction relation that is defined relative to an observation rather than to a single execution. Intuitively, a formula holds under an observation if it holds for all executions of the system that are compatible with that observation.

Definition 2.17 (Satisfaction under observation)

Let $\mathcal{S}$ be a system, $a \in Ag$, and φ a formula. Let $u = \sigma_1 \ldots \sigma_k \in Obs_a(\mathcal{S})$. We write

$$\mathcal{S}, u \models_a \varphi$$

if, for all $w \in L(\mathcal{S})$ such that $obs_a(w_{\leq k}) = u$, we have

$$\mathcal{S}, w, k \models \varphi.$$

Proposition 2.18

Let $\mathcal{S}$ be a system, $a \in Ag$, and φ be an LTL$^\sim$ formula. Let $w \in L(\mathcal{S})$ and $k \geq 1$, and set $u = obs_a(w_{\leq k})$. Then,

$$\mathcal{S}, u \models_a \varphi \quad \iff \quad \mathcal{S}, w, k \models \mathsf{K}_a\varphi\,.$$

Proof. Let $u = obs_a(w_{\leq k})$.

($\Longrightarrow$) Assume $\mathcal{S}, u \models_a \varphi$. Let $w' \in L(\mathcal{S})$ be such that $(w,k) \sim_a (w',k)$. By definition of indistinguishability (perfect recall),

$$(w,k) \sim_a (w',k) \quad \iff \quad obs_a(w'_{\leq k}) = obs_a(w_{\leq k}) = u.$$

Hence $obs_a(w'_{\leq k}) = u$, and by the definition of $\models_a$, we obtain $\mathcal{S}, w', k \models \varphi$. Since this holds for all such w', we conclude $\mathcal{S}, w, k \models \mathsf{K}_a\varphi$.

($\Longleftarrow$) Assume $\mathcal{S}, w, k \models \mathsf{K}_a\varphi$. Let $w' \in L(\mathcal{S})$ with $obs_a(w'_{\leq k}) = u$. Then $(w,k) \sim_a (w',k)$, again by the definition of indistinguishability. By the semantics of knowledge, this implies $\mathcal{S}, w', k \models \varphi$. Since this holds for every w' with $obs_a(w'_{\leq k}) = u$, we conclude $\mathcal{S}, u \models_a \varphi$. □

2.5 The Model-Checking Problem

We now turn to the algorithmic problem of deciding whether a system satisfies a given epistemic temporal specification. It serves as a basis for the construction of monitors and verification procedures.

Definition 2.19 (Model-checking problem)

Given a Büchi automaton $\mathcal{S} = (Q, \iota, \Delta, F)$ and an LTL$^\sim$ formula φ, the *model-checking problem* asks whether

$$\mathcal{S} \models \varphi,$$

that is, whether for every $w \in L(\mathcal{S})$, we have $\mathcal{S}, w, 1 \models \varphi$.

LTL model-checking problems are usually be solved by translating the negation of φ into another Büchi automaton and checking the product for emptiness. Due to the knowledge operator, unlike the classical construction, this Büchi automaton $\mathcal{A}^{\mathcal{S}}_{\neg\varphi}$ also depends on the given system $\mathcal{S}$.

To make this explicit, we introduce a transducer that combines the behavior of the system with the evaluation of the formula along its executions. That is, it maps each position

along an execution to the truth value of the formula at that position.

Definition 2.20 (System-formula transducer)

Let $\mathcal{S}$ be a Büchi automaton over Σ (representing a system) and φ be an LTL$^{\sim}$ formula. An $(\mathcal{S}, \varphi)$-*transducer* consists of

- a Büchi automaton $\mathcal{A}_\varphi^{\mathcal{S}} = (Q_\varphi, \iota_\varphi, \Delta_\varphi, F_\varphi)$ over Σ, and
- a mapping $\gamma_\varphi \colon Q_\varphi \to \{0, 1\}$

such that $L(\mathcal{A}_\varphi^{\mathcal{S}}) = L(\mathcal{S})$ and, for all $w \in L(\mathcal{S})$, all accepting runs $\pi = q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} \dots$ such that $\mathit{trace}(\pi) = w$, and all $k \in \mathbb{N}_{\geq 1}$, we have

$$\gamma_\varphi(q_k) = 1 \quad \iff \quad \mathcal{S}, w, k \models \varphi.$$

Intuitively, γ_φ marks the positions that satisfy φ.

Theorem 2.21 (Transducer construction)

For a Büchi automaton $\mathcal{S}$ and an LTL$^{\sim}$ formula φ, we can effectively construct an $(\mathcal{S}, \varphi)$-transducer.

In the construction, we will actually make use of a *monitor*:

Definition 2.22 (Knowledge monitor)

Let $\mathcal{S}$ be a Büchi automaton over Σ, an agent $a \in Ag$, and φ be an LTL$^{\sim}$ formula. An $(\mathcal{S}, \varphi)$-*knowledge monitor for* a is a DFA $\mathcal{K}_\varphi^{\mathcal{S},a}$ over Σ_a such that, for all finite words $u \in \mathit{Obs}_a(\mathcal{S}) \subseteq \Sigma_a^+$,

$$u \in L(\mathcal{K}_\varphi^{\mathcal{S},a}) \quad \iff \quad \mathcal{S}, u \models_a \varphi.$$

Theorem 2.23 (Monitor construction)

Given a Büchi automaton over Σ, an agent $a \in Ag$, and an LTL$^{\sim}$ formula φ, we can effectively construct an $(\mathcal{S}, \varphi)$-knowledge monitor for a.

The rest of the section is devoted to the simultaneous proof of Theorems 2.21 and 2.23. We construct $\mathcal{A}_\varphi^{\mathcal{S}}$ and γ_φ by structural induction on φ. We asume that $L(\mathcal{S}) \neq \emptyset$ because, otherwise, the construction is trivial.

Base case: $\varphi = p$ where $p \in AP$

We have $\mathcal{S}, w, k \models p$ iff $p \in \sigma_k$.

Assume $\mathcal{S} = (Q_{\mathcal{S}}, \iota_{\mathcal{S}}, \Delta_{\mathcal{S}}, F_{\mathcal{S}})$.

We construct the Büchi automaton:

$$\mathcal{A}_p^{\mathcal{S}} = (Q, \iota, \Delta, F)$$

where:

- $Q = Q_{\mathcal{S}} \times \{0, 1\}$
- $\iota = (\iota_{\mathcal{S}}, 0)$ (second component arbitrary)
- $\Delta = \big\{((q, \theta), \sigma, (q', \theta')) \mid (q, \sigma, q') \in \Delta_{\mathcal{S}} \text{ and } \theta' = 1 \iff p \in \sigma\big\}$
- $F = F_{\mathcal{S}} \times \{0, 1\}$

The output function is given by $\gamma_p((q, \theta) = \theta$.

The automaton tracks $\mathcal{S}$ in the first component and stores in the second component whether p holds.

Negation: $\varphi = \neg\psi$

Assume we have constructed:

$$\mathcal{A}_\psi^{\mathcal{S}} = (Q_\psi, \iota_\psi, \Delta_\psi, F_\psi)$$

with output function $\gamma_\psi \colon Q_\psi \to \{0, 1\}$. We have $\mathcal{S}, w, k \models \neg\psi$ iff $\mathcal{S}, w, k \not\models \psi$.

We construct:

$$\mathcal{A}_{\neg\psi}^{\mathcal{S}} = (Q_\psi, \iota_\psi, \Delta_\psi, F_\psi)$$

with output function $\gamma_{\neg\psi}(q) = 1 - \gamma_\psi(q)$.

The automaton is identical to $\mathcal{A}_\psi^{\mathcal{S}}$; only the output function is flipped.

Conjunction: $\varphi = \psi_1 \wedge \psi_2$

Assume we have constructed:

$$\mathcal{A}_{\psi_1}^{\mathcal{S}} = (Q_1, \iota_1, \Delta_1, F_1) \text{ with } \gamma_{\psi_1} : Q_1 \to \{0, 1\}$$
$$\mathcal{A}_{\psi_2}^{\mathcal{S}} = (Q_2, \iota_2, \Delta_2, F_2) \text{ with } \gamma_{\psi_2} : Q_2 \to \{0, 1\}$$

We have $\mathcal{S}, w, k \models \psi_1 \wedge \psi_2$ iff $\mathcal{S}, w, k \models \psi_1$ and $\mathcal{S}, w, k \models \psi_2$.

We first construct the generalized Büchi automaton

$$\mathcal{A} = (Q, \iota, \Delta, \mathcal{F})$$

where:

- $Q = Q_1 \times Q_2$
- $\iota = (\iota_1, \iota_2)$
- $\Delta = \big\{((q_1, q_2), \sigma, (q_1', q_2')) \mid (q_1, \sigma, q_1') \in \Delta_1 \text{ and } (q_2, \sigma, q_2') \in \Delta_2\big\}$
- $\mathcal{F} = \{F_1 \times Q_2,\, Q_1 \times F_2\}$

The output function is given by $\gamma((q_1, q_2)) = \gamma_{\psi_1}(q_1) \wedge \gamma_{\psi_2}(q_2)$.

We then convert this GBA to a standard Büchi automaton to obtain $\mathcal{A}^{\mathcal{S}}_{\psi_1 \wedge \psi_2}$, extending the output function accordingly during the conversion to obtain $\gamma_{\psi_1 \wedge \psi_2}$.

Strict since: $\varphi = \psi_1 \mathsf{S}^+ \psi_2$

Assume we have constructed

$$\mathcal{A}^{\mathcal{S}}_{\psi_i} = (Q_i, \iota_i, \Delta_i, F_i)$$

with output maps $\gamma_{\psi_i} : Q_i \to \{0, 1\}$ for $i \in \{1, 2\}$.

We construct a generalized Büchi automaton

$$\mathcal{A} = (Q, \iota, \Delta, \mathcal{F})$$

where:

- $Q = Q_1 \times Q_2 \times \{\mathit{init}, 0, 1\}$
- $\iota = (\iota_1, \iota_2, \mathit{init})$
- Δ contains $((q_1, q_2, \theta), \sigma, (q_1', q_2', \theta'))$ if $(q_1, \sigma, q_1') \in \Delta_1$, $(q_2, \sigma, q_2') \in \Delta_2$, and

$$\theta' = \begin{cases} 0 & \text{if } \theta = \mathit{init} \\ 1 & \text{if } \theta \in \{0, 1\} \text{ and } \big(\gamma_{\psi_2}(q_2) = 1 \text{ or } (\gamma_{\psi_1}(q_1) = 1 \text{ and } \theta = 1)\big) \\ 0 & \text{otherwise} \end{cases}$$

Intuition. After reading σ_k, the component states q_1 and q_2 correspond to position k (as in the base-case construction), and $\theta \in \{0, 1\}$ stores the truth value of $\psi_1 \mathsf{S}^+ \psi_2$ at

position k. At the very first step, strict since is always false (no previous position), so we force $\theta' = 0$ when $\theta = \mathit{init}$.

For subsequent steps, the update implements the following recurrence:

$$\mathcal{S}, w, k{+}1 \models \psi_1 \mathsf{S}^+ \psi_2 \iff \big(\mathcal{S}, w, k \models \psi_2\big) \text{ or } \big(\mathcal{S}, w, k \models \psi_1 \text{ and } \mathcal{S}, w, k \models \psi_1 \mathsf{S}^+ \psi_2\big)$$

Acceptance condition. The acceptance condition only has to ensure that the components for ψ_1 and ψ_2 are accepting on their own:

$$\mathcal{F} = \big\{\ F_1 \times Q_2 \times \{0,1\},\ Q_1 \times F_2 \times \{0,1\}\ \big\}$$

Output map. Accordingly, we define the output map by:

$$\gamma((q_1, q_2, \theta)) = \begin{cases} 0 & \text{if } \theta = \mathit{init} \\ \theta & \text{if } \theta \in \{0,1\} \end{cases}$$

Finally, we transform the GBA $\mathcal{A}$ into an ordinary Büchi automaton to obtain $\mathcal{A}^{\mathcal{S}}_{\psi_1 \mathsf{S}^+ \psi_2}$, and we accordingly adjust γ to obtain $\gamma_{\psi_1 \mathsf{S}^+ \psi_2}$ as required.

Strict until: $\varphi = \psi_1 \mathsf{U}^+ \psi_2$

Assume we have constructed

$$\mathcal{A}^{\mathcal{S}}_{\psi_i} = (Q_i, \iota_i, \Delta_i, F_i)$$

with output maps $\gamma_{\psi_i} : Q_i \to \{0,1\}$ for $i \in \{1,2\}$.

Similarly to strict since, we construct a generalized Büchi automaton

$$\mathcal{A} = (Q, \iota, \Delta, \mathcal{F})$$

where:

- $Q = Q_1 \times Q_2 \times \{\mathit{init}, 0, 1\}$
- $\iota = (\iota_1, \iota_2, \mathit{init})$
- Δ contains $((q_1, q_2, \theta), \sigma, (q_1', q_2', \theta'))$ if $(q_1, \sigma, q_1') \in \Delta_1$, $(q_2, \sigma, q_2') \in \Delta_2$, and the following hold:
  - $\theta' \in \{0,1\}$
  - $\theta \in \{0,1\} \implies \big(\theta = 1 \iff \gamma_{\psi_2}(q_2') = 1 \text{ or } (\gamma_{\psi_1}(q_1') = 1 \text{ and } \theta' = 1)\big)$

Intuition. The transition constraints implement the one-step unfolding of strict until. A positive guess $\theta = 1$ claims that ψ_2 will occur at some strictly future position and that ψ_1 holds until then (starting from the next position). As long as ψ_2 does not hold, the automaton must therefore preserve $\theta = 1$ and require ψ_1 to hold at the next position. A negative guess $\theta = 0$ claims that no such future position exists. Such a guess may persist as long as ψ_2 does not occur, but it is refuted as soon as ψ_2 holds at the next position.

Formally, the transition constraints enforce the following semantic recurrence:

$$\mathcal{S}, w, k \models \psi_1 \mathsf{U}^+ \psi_2 \iff \big(\mathcal{S}, w, k{+}1 \models \psi_2\big) \text{ or } \big(\mathcal{S}, w, k{+}1 \models \psi_1 \text{ and } \mathcal{S}, w, k{+}1 \models \psi_1 \mathsf{U}^+ \psi_2\big)$$

Acceptance condition. The acceptance condition has to ensure that positive guesses cannot remain unresolved forever. In particular, an accepting run cannot stay in states with $\theta = 1$ infinitely often without eventually visiting a position where ψ_2 holds. Therefore, every accepting run resolves each positive guess by a strictly future ψ_2-position. This is achieved using

$$\mathcal{F} = \big\{\ F_1 \times Q_2 \times \{0,1\},\ Q_1 \times F_2 \times \{0,1\},\ \{(q_1, q_2, \theta) \mid \theta = 0 \text{ or } \gamma_{\psi_2}(q_2) = 1\}\ \big\}.$$

Output map. Accordingly, we define the output map by:

$$\gamma((q_1, q_2, \theta)) = \begin{cases} 0 & \text{if } \theta = \mathit{init} \\ \theta & \text{if } \theta \in \{0,1\} \end{cases}$$

Finally, we obtain the required $\mathcal{A}^{\mathcal{S}}_{\psi_1 \mathsf{U}^+ \psi_2}$ and the adjusted $\gamma_{\psi_1 \mathsf{U}^+ \psi_2}$ transforming $\mathcal{A}$ into a Büchi automaton.

Knowledge monitor:

A knowledge monitor does not observe the actual events of the system, but only the partial observations available to a given agent. As a consequence, after each observation, the monitor cannot determine a single current system state. Instead, it maintains a *belief set* consisting of all system states that are consistent with the observations seen so far. As the observation sequence grows, this belief set is updated accordingly. The monitor evaluates a property by checking whether it holds in all states of the current belief set. In this way, the monitor can determine when a property is known to hold and when it cannot yet be concluded from the available observations.

Assume we have constructed

$$\mathcal{A}^{\mathcal{S}}_{\varphi} = (Q_\varphi, \iota_\varphi, \Delta_\varphi, F_\varphi)$$

with output function $\gamma_\varphi : Q_\varphi \to \{0,1\}$.

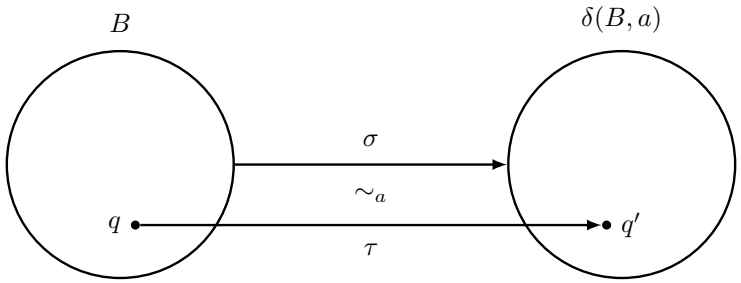


Figure 2.1: Belief update and representative state transition.

Without loss of generality, we assume that $\mathcal{A}_\varphi^{\mathcal{S}}$ is *productive*, that is, for every state q reachable from ι_φ, there is an accepting q-run. Otherwise, $\mathcal{A}_\varphi^{\mathcal{S}}$ can be replaced by an equivalent productive automaton obtained by removing all states from which there is no accepting run.

Given $a \in Ag$, we construct a $(\mathcal{S}, \varphi)$-knowledge monitor for a, a DFA $\mathcal{K}_\varphi^{\mathcal{S},a}$ over Σ_a, as

$$\mathcal{K}_\varphi^{\mathcal{S},a} = (Q, \iota, \delta, F)$$

where:

- $Q = 2^{Q_\varphi}$ (belief states)
- $\iota = \{\iota_\varphi\}$
- δ is defined by

$$\delta(B,\sigma) = \left\{ q' \in Q_\varphi \;\middle|\; \exists q \in B\colon \exists \tau \in \Sigma\colon (q,\tau,q') \in \Delta_\varphi \text{ and } \sigma = obs_a(\tau) \right\}$$

  (illustrated in Figure 2.1)
- $F = \{B \in 2^{Q_\varphi} \mid \forall q \in B\colon \gamma_\varphi(q) = 1\}$

Correctness of monitor. We have to show that, for all $u \in Obs_a(\mathcal{S})$,

$$\delta^*(\iota, u) \in F \quad \Longleftrightarrow \quad \mathcal{S}, u \models_a \varphi. \tag{2.1}$$

Key invariant (belief semantics). For every $u \in \Sigma_a^*$,

$$\delta^*(\iota,u) = \left\{ q \in Q_\varphi \;\middle|\; \begin{array}{l} \exists v \in \Sigma^*\colon u \sim_a v \text{ and} \\ \exists \pi \text{ a finite run of } \mathcal{A}_\varphi^{\mathcal{S}} \text{ over } v \text{ with } \pi(|v|) = q \end{array} \right\}. \tag{2.2}$$

Proof of (2.2). We proceed by induction on $k = |u|$.

Base $u = \varepsilon$. ($\subseteq$) We have $\delta^*(\iota, u) = \{\iota_\varphi\}$. Taking $v = \varepsilon$ yields $u \sim_a v$. Moreover, the trivial run $\pi = \iota_\varphi$ is a finite run over v with $\pi(0) = \iota_\varphi$.

($\supseteq$) Conversely, let q belong to the right-hand side of (2.2). The unique v such that $u \sim_a v$ is $v = \varepsilon$. Over ε, the only finite run is the trivial run $\pi = \iota_\varphi$, hence $q = \pi(0) = \iota_\varphi$.

Step $u \to u\sigma$. Let $u \in \Sigma_a^*$ and $\sigma \in \Sigma_a$. Let $k = |u|$.

($\subseteq$) Let $q' \in \delta^*(\iota, u\sigma)$. By definition of the belief update, there exist $q \in \delta^*(\iota, u)$ and a letter $\tau \in \Sigma$ such that

$$\sigma = obs_a(\tau) \quad \text{and} \quad (q, \tau, q') \in \Delta_\varphi.$$

By the induction hypothesis for u, there exist $v \in \Sigma^*$ with $u \sim_a v$ and a finite run $\hat{\pi}$ over v such that $\hat{\pi}(k) = q$. Let π be obtained from $\hat{\pi}$ by appending the transition $q \xrightarrow{\tau} q'$. Then, π is a finite run over $v\tau$ and satisfies $\pi(k+1) = q'$. Moreover, from $u \sim_a v$ and $\sigma = obs_a(\tau)$, we get $u\sigma \sim_a v\tau$. Hence q' belongs to the right-hand side.

($\supseteq$) Conversely, assume $q' \in Q_\varphi$ belongs to the right-hand side for $u\sigma$. Then, there exist $v = \tau_1 \dots \tau_k \in \Sigma^*$ and $\tau \in \Sigma$ with $u\sigma \sim_a v\tau$ and a finite run π over $v\tau$ such that $\pi(k+1) = q'$. From $u\sigma \sim_a v\tau$, it follows that $u \sim_a v$, and the prefix of π up to position k is a finite run over v reaching $\pi(k)$. By the induction hypothesis, this implies $\pi(k) \in \delta^*(\iota, u)$. Moreover, $\pi(k) \xrightarrow{\tau} q'$ is a transition of Δ_φ and $\sigma = obs_a(\tau)$. Hence, by the definition of the belief update, we obtain $q' \in \delta^*(\iota, u\sigma)$.

This establishes (2.2) for all $u \in \Sigma_a^*$. □

Let $u \in Obs_a(\mathcal{S})$ and let $k = |u|$. We show (2.1).

Let $B_u = \delta^*(\iota, u)$.

($\Rightarrow$) Assume $B_u \in F$. By definition of F, this means

$$\forall q \in B_u \colon \gamma_\varphi(q) = 1.$$

Let $w \in L(\mathcal{S})$ with $obs_a(w_{\leq k}) = u$. Fix any accepting run π of $\mathcal{A}_\varphi^{\mathcal{S}}$ over w and set $q = \pi(k)$. Then, the prefix of π up to time k is a finite run over $w_{\leq k}$ reaching q. Since $obs_a(w_{\leq k}) = u$, by invariant (2.2) we have $q \in B_u$, hence $\gamma_\varphi(q) = 1$. By correctness of the inductive construction for φ, this implies $\mathcal{S}, w, k \models \varphi$. Since w was arbitrary among executions compatible with u, we conclude $\mathcal{S}, u \models_a \varphi$.

($\Leftarrow$) Assume $\mathcal{S}, u \models_a \varphi$. We show that $B_u \in F$, i.e.,

$$\forall q \in B_u \colon \ \gamma_\varphi(q) = 1.$$

So fix an arbitrary $q \in B_u$. By the invariant (2.2), there exist a word $v \in \Sigma^*$ with $obs_a(v) = u$ and a finite run π of $\mathcal{A}_\varphi^{\mathcal{S}}$ over v such that $\pi(k) = q$. Since $\mathcal{A}_\varphi^{\mathcal{S}}$ is productive, the finite run π can be extended to an accepting infinite run π' of $\mathcal{A}_\varphi^{\mathcal{S}}$. Let $w = trace(\pi')$. Then $w \in L(\mathcal{A}_\varphi^{\mathcal{S}}) = L(\mathcal{S})$ and $w_{\leq k} = v$, hence $obs_a(w_{\leq k}) = u$.

From $\mathcal{S}, u \models_a \varphi$ and $obs_a(w_{\leq k}) = u$, it follows by definition of $\models_a$ that $\mathcal{S}, w, k \models \varphi$. By correctness of the inductive construction for φ, this implies $\gamma_\varphi(q) = 1$.

Since $q \in B_u$ was arbitrary, we conclude that $B_u \in F$.

This proves (2.1) and hence the correctness of the monitor.

Knowledge operator: $\varphi = \mathsf{K}_a\psi$

Assume we have constructed the $(\mathcal{S}, \psi)$-knowledge monitor $\mathcal{K}_\psi^{\mathcal{S},a} = (Q_\mathcal{K}, \iota_\mathcal{K}, \delta_\mathcal{K}, F_\mathcal{K})$. We construct the product of $\mathcal{K}_\psi^{\mathcal{S},a}$ with $\mathcal{S} = (Q_\mathcal{S}, \iota_\mathcal{S}, \Delta_\mathcal{S}, F_\mathcal{S})$:

$$\mathcal{A}^\mathcal{S}_{\mathsf{K}_a\psi} = (Q, \iota, \Delta, F)$$

where:

- $Q = Q_\mathcal{K} \times Q_\mathcal{S} = 2^{Q_\psi} \times Q_\mathcal{S}$
- $\iota = (\iota_\mathcal{K}, \iota_\mathcal{S})$
- Δ contains the triples $((B, q), \sigma, (B', q'))$ such that $\delta_\mathcal{K}(B, obs_a(\sigma)) = B'$ and $(q, \sigma, q') \in \Delta_\mathcal{S}$
- $F = Q_\mathcal{K} \times F_\mathcal{S}$ (acceptance determined by $\mathcal{S}$ component)

The output function is defined as follows:

$$\gamma_{\mathsf{K}_a\psi}((B, q)) = \begin{cases} 1 & \text{if } B \in F_\mathcal{K} \\ 0 & \text{otherwise} \end{cases}$$

Clearly, as acceptance is determined entirely by the second component, this ensures $L(\mathcal{A}^\mathcal{S}_{\mathsf{K}_a\psi}) = L(\mathcal{S})$.

Correctness of the output map. We show that for every word $w = \sigma_1\sigma_2 \ldots \in L(\mathcal{S})$, every accepting run

$$(B_0, q_0) \xrightarrow{\sigma_1} (B_1, q_1) \xrightarrow{\sigma_2} (B_2, q_2) \xrightarrow{\sigma_3} \cdots$$

of $\mathcal{A}^\mathcal{S}_{\mathsf{K}_a\psi}$ over w, and every $k \in \mathbb{N}_{\geq 1}$, we have

$$\gamma_{\mathsf{K}_a\psi}((B_k, q_k)) = 1 \quad \Longleftrightarrow \quad \mathcal{S}, w, k \models \mathsf{K}_a\psi.$$

By construction of Δ, the first component is updated according to

$$B_k = \delta^*_\mathcal{K}(\iota_\mathcal{K},\ obs_a(w_{\leq k})),$$

since at step i the monitor reads exactly the observation letter $obs_a(\sigma_i)$.

($\Rightarrow$) Assume $\gamma_{\mathsf{K}_a\psi}((B_k, q_k)) = 1$. Then $B_k \in F_{\mathcal{K}}$ by definition of $\gamma_{\mathsf{K}_a\psi}$. Let $u = obs_a(w_{\leq k})$, which is contained in $Obs_a(\mathcal{S})$. By the correctness of the knowledge-monitor construction, $u \in L(\mathcal{K}_\psi^{\mathcal{S},a})$ implies

$$\mathcal{S}, u \models_a \psi.$$

By Proposition 2.18, we obtain $\mathcal{S}, w, k \models \mathsf{K}_a\psi$.

($\Leftarrow$) Assume $\mathcal{S}, w, k \models \mathsf{K}_a\psi$. Let $u = obs_a(w_{\leq k})$. By Proposition 2.18, we have $\mathcal{S}, u \models_a \psi$. By correctness of the monitor construction, this implies $u \in L(\mathcal{K}_\psi^{\mathcal{S},a})$, hence $\delta_{\mathcal{K}}^*(\iota_{\mathcal{K}}, u) \in F_{\mathcal{K}}$. Using $B_k = \delta_{\mathcal{K}}^*(\iota_{\mathcal{K}}, u)$, we obtain $B_k \in F_{\mathcal{K}}$, and therefore $\gamma_{\mathsf{K}_a\psi}((B_k, q_k)) = 1$.

This concludes the proof of correctness of the output map for all $k \in \mathbb{N}_{\geq 1}$.

Altogether, we have shown Theorems 2.21 and 2.23.

Solving the model-checking problem

We now show how the model-checking problem can be solved using a transducer. Recall from Definition 2.19 that, given a Büchi automaton $\mathcal{S} = (Q, \iota, \Delta, F)$ and an LTL$^\sim$ formula φ, the model-checking problem asks whether $\mathcal{S} \models \varphi$, that is, whether $\mathcal{S}, w, 1 \models \varphi$ holds for every $w \in L(\mathcal{S})$.

Theorem 2.24 (Decidability of model checking)

The model-checking problem is decidable.

Proof. We reduce model checking to the emptiness problem for Büchi automata. From $\mathcal{S}$ and φ, we construct the $(\mathcal{S}, \varphi)$-transducer $\mathcal{A}_\varphi^{\mathcal{S}}$ with output map γ_φ.

Recall that a run of $\mathcal{A}_\varphi^{\mathcal{S}}$ has the form $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} \cdots$. The state q_0 is the initial state (before reading the word), and q_1 corresponds to the first letter σ_1. By correctness of the transducer construction, for every accepting run π over $w \in L(\mathcal{S})$ we have

$$\gamma_\varphi(\pi(1)) = 1 \quad \Longleftrightarrow \quad \mathcal{S}, w, 1 \models \varphi.$$

Hence, we have the following equivalences:

$$\begin{aligned} \mathcal{S} \models \varphi \quad &\Longleftrightarrow \quad \forall w \in L(\mathcal{S}) \colon \mathcal{S}, w, 1 \models \varphi \\ &\Longleftrightarrow \quad \neg\exists w \in L(\mathcal{S}) \colon \mathcal{S}, w, 1 \not\models \varphi \\ &\Longleftrightarrow \quad \neg\exists \text{ accepting run } \pi \text{ of } \mathcal{A}_\varphi^{\mathcal{S}} \text{ such that } \gamma_\varphi(\pi(1)) = 0 \end{aligned}$$

Thus, model checking reduces to checking whether $\mathcal{A}_\varphi^{\mathcal{S}}$ admits an accepting run whose state q at position 1 satisfies $\gamma_\varphi(q) = 0$. This is an emptiness problem for Büchi automata, which

can be solved using standard graph-theoretic algorithms for finding reachable accepting cycles. □

Note (Complexity)

Model checking for LTL (without knowledge operators) is PSPACE-complete. In contrast, extending LTL with epistemic operators such as K_a leads, in general, to a non-elementary complexity blow-up. This increase is caused by the construction of the knowledge monitor, which relies on a belief-state construction under perfect recall and induces an iterated powerset blow-up.

2.6 Related Work and Further Reading

Epistemic temporal logics under synchronous perfect recall have been studied extensively. Early decidability and complexity results were obtained by van der Meyden and Shilov, who identified a non-elementary blow-up inherent to reasoning about agents' information sets [12].

In more recent work, Bozzelli, Maubert, and Murano gave a precise complexity analysis in terms of the alternation depth of epistemic operators [3]. Cohen and Lomuscio studied past-time epistemic temporal logics under synchronous perfect recall and showed that, while the general problem is non-elementary, substantially better bounds can be obtained for restricted syntactic fragments [5].

Finkbeiner and Siber established decidability of the logic considered in these lecture notes via translations to first-order formalisms for hyperproperties [8]. They also considered additional operators, such as counterfactuals, that go beyond the logic treated here.

For background on epistemic logic and the semantics of knowledge, we refer to the textbook by Fagin, Halpern, Moses, and Vardi, *Reasoning about Knowledge* [7].

3

Diagnosability, Opacity, Monitoring

This chapter illustrates how epistemic temporal logic can be used to formalize and analyze classical runtime-verification properties. We focus on diagnosability and related notions, which capture whether agents can infer the occurrence or absence of faults from partial observations. Thanks to the epistemic perspective, these properties can be expressed uniformly as logical formulas. However, it should be noted that this uniform logical formulation does not, in general, yield optimal complexity bounds for checking these properties.

3.1 Diagnosability

While classical definitions of (co-)diagnosability are phrased in terms of explicit comparisons of observation projections (e.g., [11, 13]), the epistemic formulation captures the same requirement at the semantic level, expressing directly that an agent can infer the absence of an error after a bounded delay.

3.1.1 Positive and Negative Diagnosability

In the following, the prefix P refers to *positive* diagnosability, while N refers to *negative* diagnosability.

Definition 3.1 (P- and N-diagnosability)

Let $\mathcal{S}$ be a Büchi automaton over Σ, $a \in Ag$ be an agent, $e \in AP \setminus AP_a$ be an error proposition, and $D \in \mathbb{N}$ be a maximal delay. We say that $\mathcal{S}$ is

- *D-P-diagnosable* (wrt. a and e) if

$$\mathcal{S} \models \mathsf{G}\Big(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{P} e\Big),$$

- *D-N-diagnosable* (wrt. a and e) if

$$\mathcal{S} \models \mathsf{X}^D \mathsf{G}\Big(\mathsf{H}\neg e \to \mathsf{K}_a \mathsf{Y}^D \mathsf{H}\neg e\Big).$$

We say that $\mathcal{S}$ is *P-diagnosable* (*N-diagnosable*) if there is $D \in \mathbb{N}$ such that it is P-diagnosable (N-diagnosable, respectively).

Finally, we call $\mathcal{S}$ *unbounded-P-diagnosable* (wrt. a and e) if

$$\mathcal{S} \models \mathsf{G}\Big(e \to \mathsf{FK}_a \mathsf{P} e\Big).$$

Intuitively, positive diagnosability means that whenever an error occurs, agent a knows within D steps that an error has occurred in the past. In unbounded P-diagnosability, no bound is imposed on the delay before such detection.

Dually, negative diagnosability means that after an initial delay of D time steps, whenever the system has remained fault-free, agent a can conclude that the system was fault-free up to D steps in the past.

Example 3.2 (Positive- and negative-diagnosability)

We revisit Example 1.1, where $AP = \{p, r, e\}$, $a \in Ag$ is the agent in question, and e is the unobservable error proposition.

Consider again four Büchi automata $\mathcal{S}_1$–$\mathcal{S}_4$ such that:

$$\begin{aligned} L(\mathcal{S}_1) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}\{r\}^\omega \\ L(\mathcal{S}_2) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}^\omega \\ L(\mathcal{S}_3) &= \{p\}^\omega + \{p\}^*\{e\}\{p\}^\omega \\ L(\mathcal{S}_4) &= \{p\}^\omega + \{p\}^*\{p,e\}\{p\}^*\{r\}^\omega \end{aligned}$$

Then, the following hold:

- $\mathcal{S}_1$ is both 2-P-diagnosable and 2-N-diagnosable.
- $\mathcal{S}_2$ is neither P-diagnosable nor N-diagnosable.
- $\mathcal{S}_3$ is both 0-P-diagnosable and 0-N-diagnosable.
- $\mathcal{S}_4$ is neither P-diagnosable nor N-diagnosable; however, it is unbounded-P-diagnosable.

It is not a coincidence that positive- and negative-diagnosability coincide on this example. In fact, the two notions are equivalent, reflecting the duality between detecting the occurrence and the absence of errors under partial observation.

Proposition 3.3

Fix $D \in \mathbb{N}$. Let $\mathcal{S}$ be a system and fix an agent and an error proposition. Then,

$$\mathcal{S} \text{ is } D\text{-P-diagnosable} \iff \mathcal{S} \text{ is } D\text{-N-diagnosable.}$$

Proof. Fix $D \in \mathbb{N}$. Let

$$(\mathrm{P}_D) \quad \mathcal{S} \models \mathsf{G}\Big(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{P} e\Big) \qquad (\mathrm{N}_D) \quad \mathcal{S} \models \mathsf{X}^D \mathsf{G}\Big(\mathsf{H}\neg e \to \mathsf{K}_a \mathsf{Y}^D \mathsf{H}\neg e\Big).$$

We show $(\mathrm{P}_D) \Rightarrow (\mathrm{N}_D)$ and $(\mathrm{N}_D) \Rightarrow (\mathrm{P}_D)$.

$(\mathrm{P}_D) \Rightarrow (\mathrm{N}_D)$. Assume (P_D). We prove (N_D), i.e. we show that for all $w \in L(\mathcal{S})$ and all positions $k \geq 1$,

$$\mathcal{S}, w, k{+}D \models \mathsf{H}\neg e \implies \mathcal{S}, w, k{+}D \models \mathsf{K}_a \mathsf{Y}^D \mathsf{H}\neg e.$$

So fix $w \in L(\mathcal{S})$ and $k \geq 1$ with $\mathcal{S}, w, k{+}D \models \mathsf{H}\neg e$. Let $w' \in L(\mathcal{S})$ with $(w, k{+}D) \sim_a (w', k{+}D)$. It suffices to prove $\mathcal{S}, w', k{+}D \models \mathsf{Y}^D \mathsf{H}\neg e$, equivalently $\mathcal{S}, w', k \models \mathsf{H}\neg e$.

Suppose towards a contradiction that $\mathcal{S}, w', k \not\models \mathsf{H}\neg e$. Then $\mathcal{S}, w', k \models \mathsf{P} e$, so choose j with $1 \leq j \leq k$ such that $\mathcal{S}, w', j \models e$. By (P_D) applied to w' at time j,

$$\mathcal{S}, w', j{+}D \models \mathsf{K}_a \mathsf{P} e.$$

Since $(w, k{+}D) \sim_a (w', k{+}D)$ means $obs_a(w_{\leq k+D}) = obs_a(w'_{\leq k+D})$, by truncation we obtain $obs_a(w_{\leq j+D}) = obs_a(w'_{\leq j+D})$, i.e. $(w, j{+}D) \sim_a (w', j{+}D)$. By the semantics of knowledge, this yields

$$\mathcal{S}, w, j{+}D \models \mathsf{P} e.$$

As $j{+}D \leq k{+}D$, it follows that $\mathcal{S}, w, k{+}D \models \mathsf{P} e$, contradicting $\mathcal{S}, w, k{+}D \models \mathsf{H}\neg e \equiv \neg \mathsf{P} e$. Hence $\mathcal{S}, w', k \models \mathsf{H}\neg e$, as required. Therefore $\mathcal{S}, w, k{+}D \models \mathsf{K}_a \mathsf{Y}^D \mathsf{H}\neg e$, proving (N_D).

$(\mathrm{N}_D) \Rightarrow (\mathrm{P}_D)$. Assume (N_D). We prove (P_D), i.e. for all $w \in L(\mathcal{S})$ and all $k \geq 1$,

$$\mathcal{S}, w, k \models e \implies \mathcal{S}, w, k{+}D \models \mathsf{K}_a \mathsf{P} e.$$

So fix $w \in L(\mathcal{S})$ and $k \geq 1$ with $\mathcal{S}, w, k \models e$. Let $w' \in L(\mathcal{S})$ with $(w, k{+}D) \sim_a (w', k{+}D)$. We show $\mathcal{S}, w', k{+}D \models \mathsf{P} e$.

Suppose towards a contradiction that $\mathcal{S}, w', k{+}D \not\models \mathsf{P} e$. Then $\mathcal{S}, w', k{+}D \models \mathsf{H}\neg e$. Apply (N_D) to w' at time k:

$$\mathcal{S}, w', k{+}D \models \mathsf{K}_a \mathsf{Y}^D \mathsf{H}\neg e.$$

Since $(w, k{+}D) \sim_a (w', k{+}D)$, the semantics of knowledge yields

$$\mathcal{S}, w, k{+}D \models \mathsf{Y}^D \mathsf{H} \neg e,$$

i.e. $\mathcal{S}, w, k \models \mathsf{H} \neg e$, contradicting $\mathcal{S}, w, k \models e$. Hence $\mathcal{S}, w', k{+}D \models \mathsf{P} e$.

Since w' was arbitrary, $\mathcal{S}, w, k{+}D \models \mathsf{K}_a \mathsf{P} e$. Thus (P_D) holds. □

Exercise 3.4 (Four alternative definitions of diagnosability)

Let $\mathcal{S}$ be a system, $a \in Ag$ be an agent, and $e \in AP \setminus AP_a$ be an unobservable error proposition. Fix a delay bound $D \in \mathbb{N}$.

Consider the following alternative candidate formulas for diagnosability:

$$\varphi_1 = \mathsf{G}\Big(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{Y}^D \mathsf{P} e\Big)$$
$$\varphi_2 = \mathsf{G}\Big(\mathsf{H} \neg e \to \mathsf{X}^D \mathsf{K}_a \mathsf{Y}^D \mathsf{H} \neg e\Big)$$
$$\varphi_3 = \mathsf{X}^D \mathsf{G}\Big(\mathsf{H} \neg e \to \mathsf{Y}^D \mathsf{K}_a \mathsf{H} \neg e\Big)$$
$$\varphi_4 = \mathsf{G}\Big(\mathsf{H} \neg e \to \mathsf{X}^D \mathsf{K}_a \mathsf{H} \neg e\Big)$$

Give an informal interpretation of each formula and explain why neither formula captures diagnosability as we had defined it before. Which of them are "reasonable" definitions.

Hint: You may use $\{p, e\}\{p\}^\omega + \{p\}\{p, e\}\{p\}^\omega$ to separate certain formulas.

We conclude this section by considering the algorithmic complexity of diagnosability problems. Although the definitions are given in terms of epistemic temporal formulas, the corresponding decision problems can be solved efficiently on finite-state systems. We will exemplify this for D-P-diagnosability.

Fix a (system) Büchi automaton $\mathcal{S} = (Q, \iota, \Delta, F)$, an agent $a \in Ag$, an error proposition $e \in AP \setminus AP_a$, and a bound $D \in \mathbb{N}$. We say that a word $u = \sigma_1 \dots \sigma_n \in \Sigma^*$ is *faulty* if there exists $i \in \{1, \dots, n\}$ such that $e \in \sigma_i$. If u is not faulty, it is called *fault-free*.

The algorithm is based on the following characterization of D-P-diagnosability:

Proposition 3.5 (Characterization of bounded P-diagnosability)

The following statements are equivalent:

(1) $\mathcal{S}$ is *not* D-P-diagnosable (wrt. a and e), i.e. $\mathcal{S} \not\models \mathsf{G}\big(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{P} e\big)$.

(2) There exist two words $w, w' \in L(\mathcal{S})$ and a position $k \geq 1$ such that we have (i) $\mathcal{S}, w, k \models e$, (ii) $w'_{\leq k+D}$ is fault-free, and (iii) $obs_a(w_{\leq k+D}) = obs_a(w'_{\leq k+D})$.

Proof. We show (1) $\Leftrightarrow$ (2).

(1) $\Rightarrow$ (2): Assume

$$\mathcal{S} \not\models \mathsf{G}\big(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{P} e\big).$$

Then there exist $w \in L(\mathcal{S})$ and a position $k \geq 1$ such that

$$\mathcal{S}, w, k \models e \quad \text{and} \quad \mathcal{S}, w, k+D \not\models \mathsf{K}_a \mathsf{P} e.$$

According to the semantics of K_a, there exists $w' \in L(\mathcal{S})$ such that

$$obs_a(w_{\leq k+D}) = obs_a(w'_{\leq k+D}) \qquad \text{and} \qquad \mathcal{S}, w', k+D \not\models \mathsf{P} e.$$

The latter implies that $w'_{\leq k+D}$ is fault-free. Thus w, w' and k satisfy the requirements in (2).

(2) $\Rightarrow$ (1): Assume that there exist $w, w' \in L(\mathcal{S})$ and $k \geq 1$ such that $\mathcal{S}, w, k \models e$, the prefix $w'_{\leq k+D}$ is fault-free, and $obs_a(w_{\leq k+D}) = obs_a(w'_{\leq k+D})$. Since $w'_{\leq k+D}$ is fault-free, we have

$$\mathcal{S}, w', k+D \not\models \mathsf{P} e.$$

Moreover, $obs_a(w_{\leq k+D}) = obs_a(w'_{\leq k+D})$ means that $(w, k+D) \sim_a (w', k+D)$. Hence, by the semantics of knowledge,

$$\mathcal{S}, w, k+D \not\models \mathsf{K}_a \mathsf{P} e.$$

Together with $\mathcal{S}, w, k \models e$, this shows that

$$\mathcal{S} \not\models \mathsf{G}\big(e \to \mathsf{X}^D \mathsf{K}_a \mathsf{P} e\big),$$

i.e. $\mathcal{S}$ is not D-P-diagnosable. □

Theorem 3.6 (Deciding bounded P-diagnosability in polynomial time)

The following problem can be decided in polynomial time:

Given a Büchi automaton $\mathcal{S}$ over Σ, $D \in \mathbb{N}$ (encoded in unary), $a \in Ag$, and an unobservable error proposition e, is $\mathcal{S}$ D-P-diagnosable (wrt. a and e)?

Exercise 3.7

Prove Theorem 3.6 using Proposition 3.5.

Note (On unbounded-P-diagnosability)

Unbounded-P-diagnosability is not a standard notion in the classical diagnosis literature, but it arises naturally in the logical formulation. Unlike P-diagnosability, it does not guarantee a uniform bound on the delay until detection and therefore provides weaker runtime guarantees.

In finite-state systems modeled as Kripke structures (or, equivalently, automata where all states are accepting), unbounded-P-diagnosability and P-diagnosability coincide: eventual detection implies detection within a uniform bound. This fact underlies the classical diagnosability algorithms.

In contrast, in the more general setting of Büchi automata, acceptance conditions may rule out infinite indistinguishability without imposing a uniform bound. As a result, unbounded-P-diagnosability and P-diagnosability no longer coincide.

Note (General error properties)

Finally, we note that most of the diagnosability literature focuses on a distinguished fault action or fault proposition. In contrast, the epistemic-logic framework allows error events to be specified by arbitrary temporal formulas, so that one can reason about more general classes of faulty behaviors.

3.1.2 Diagnosers and Knowledge Monitors

In classical diagnosis, one often starts with an abstract definition of a diagnoser as a function that maps finite observation sequences to diagnosis decisions.

Definition 3.8 (Diagnoser)

Let $\mathcal{S}$ be a Büchi automaton, let $a \in Ag$ be an agent, let $e \in AP \setminus AP_a$ be an unobservable error proposition, and let $D \in \mathbb{N}$. A *D-P-diagnoser for $\mathcal{S}$* (wrt. a and e) is a function

$$f \colon \Sigma_a^* \to \{0, 1\}$$

such that the following conditions hold:

1. (*Detection*) For all faulty $u \in \Sigma^+$ and all $v \in \Sigma^*$ such that $|v| = D$ and $uv \in \mathit{Pref}^+(L(\mathcal{S}))$, we have $f(\mathit{obs}_a(uv)) = 1$.
2. (*No false alarms*) For all fault-free $u \in \mathit{Pref}^+(L(\mathcal{S}))$, we have $f(\mathit{obs}_a(u)) = 0$.

The first condition means that on every execution prefix u in which a fault has already occurred, the diagnoser outputs 1 after D further steps. Note that since the condition holds for all faulty prefixes, detection persists at all later times. The second condition forbids false positives.

We now relate this classical notion to knowledge monitors.

Proposition 3.9 (Knowledge monitors as diagnosers)

Let $\mathcal{S}$ be a Büchi automaton, let $a \in \mathit{Ag}$, let $e \in \mathit{AP} \setminus \mathit{AP}_a$, and let $D \in \mathbb{N}$. If $\mathcal{S}$ is D-P-diagnosable (wrt. a and e), then the function $f = [\![\mathcal{K}^{\mathcal{S},a}_{\mathsf{P}e}]\!]$ is a D-P-diagnoser for $\mathcal{S}$.

Recall that $\mathcal{K}^{\mathcal{S},a}_{\mathsf{P}e}$ is an $(\mathcal{S}, \mathsf{P}e)$-knowledge monitor and that f satisfies, for all $u \in \mathit{Obs}_a(\mathcal{S})$:

$$f(u) = \begin{cases} 1 & \text{if } \mathcal{S}, u \models_a \mathsf{P}e \\ 0 & \text{otherwise} \end{cases}$$

Proof. We show the two properties of a D-P-diagnoser.

No false alarms. Let $u \in \mathit{Pref}^+(L(\mathcal{S}))$ be fault-free. Then $\mathcal{S}, \mathit{obs}_a(u) \not\models_a \mathsf{P}e$, hence $f(\mathit{obs}_a(u)) = 0$.

Detection. Let u be faulty and let v with $|v| = D$ such that $uv \in \mathit{Pref}^+(L(\mathcal{S}))$. Choose $w \in L(\mathcal{S})$ extending uv and let $k \leq |u|$ with $\mathcal{S}, w, k \models e$. By D-P-diagnosability, $\mathcal{S}, w, k{+}D \models \mathsf{K}_a\mathsf{P}e$. Since $k{+}D \leq |uv|$, it follows that $\mathcal{S}, \mathit{obs}_a(uv) \models_a \mathsf{P}e$. By correctness of the knowledge monitor, $f(\mathit{obs}_a(uv)) = 1$.

Thus f satisfies both conditions of a D-P-diagnoser. □

Exercise 3.10

Consider running $\mathcal{K}^{\mathcal{S},a}_{\mathsf{P}e}$ and $\mathcal{K}^{\mathcal{S},a}_{\varphi}$, where $\varphi = \mathsf{Y}^D\mathsf{H}\neg e$, in parallel on the same observation stream.

1. Describe the possible combinations of verdicts and interpret them epistemically. When can a specific combination happen (can the two diagnosers accept

simultaneously)?

2. Explain why this yields a three-valued diagnosis semantics.

3.1.3 Decentralized Diagnosis

We now turn to a decentralized setting, where several agents observe the system independently and no single agent has access to all observations. In this context, diagnosability requires that the occurrence or absence of an error can be inferred by at least one agent, based solely on its own observations. This leads to the notions of positive and negative *codiagnosability*.

Definition 3.11 (P- and N-codiagnosability)

Let $\mathcal{S}$ be a Büchi automaton, $e \in AP \setminus \bigcup_{a \in Ag} AP_a$ be an error proposition, and $D \in \mathbb{N}$. We say that $\mathcal{S}$ is

- *D-P-codiagnosable* (wrt. e) if

$$\mathcal{S} \models \mathsf{G}\Big(e \to \mathsf{X}^D \bigvee_{a \in Ag} \mathsf{K}_a \mathsf{P} e\Big),$$

- *D-N-codiagnosable* (wrt. e) if

$$\mathcal{S} \models \mathsf{X}^D \mathsf{G}\Big(\mathsf{H}\neg e \to \bigvee_{a \in Ag} \mathsf{K}_a \mathsf{Y}^D \mathsf{H}\neg e\Big).$$

Accordingly, we call $\mathcal{S}$ *P-codiagnosable* (*N-codiagnosable*) if there is $D \in \mathbb{N}$ such that it is D-P-codiagnosable (D-N-codiagnosable, respectively).

Unlike in the one-agent case, the two decentralized versions of diagnosability are incomparable:

Example 3.12 (P- and N-codiagnosability are incomparable)

Let $Ag = \{a_1, a_2\}$, $AP = \{p_1, p_2, e\}$, $AP_{a_1} = \{p_1\}$, and $AP_{a_2} = \{p_2\}$. In particular, e is the unobservable error proposition.

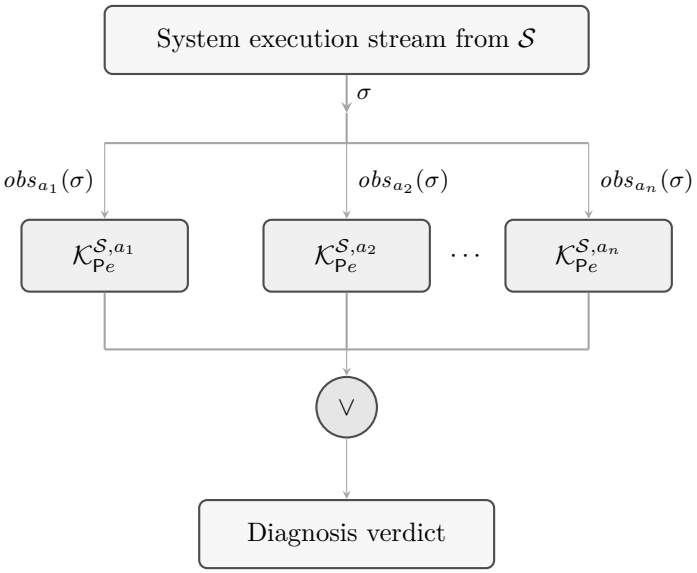


Figure 3.1: Decentralized diagnosis via local knowledge monitors (positive case). Each global event σ is locally filtered into $obs_{a_i}(\sigma)$ and processed by the corresponding monitor.

Consider Büchi automata $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ such that:

$$L(\mathcal{S}_1) = \emptyset^*\{e\}(\{p_1\} + \{p_2\})\emptyset^\omega$$
$$L(\mathcal{S}_2) = (\{p_1\} + \{p_2\} + \{e\})\emptyset^\omega$$
$$L(\mathcal{S}_3) = (\{e\}\{p_1\}\{p_2\} + \{p_2\}\{p_1\})\emptyset^\omega$$

Then,

- $\mathcal{S}_1$ is 1-P-codiagnosable but not N-codiagnosable,
- $\mathcal{S}_2$ is 0-N-codiagnosable but not P-codiagnosable,
- $\mathcal{S}_3$ is neither P-codiagnosable nor N-codiagnosable.

Note that $\mathcal{S}_3$ would be P-diagnosable with respect to an agent that can observe both p_1 and p_2, since the order in which these propositions are observed allows the agent to infer whether an error has occurred.

Knowledge monitors as decentralized diagnosers. In the decentralized setting, each agent a receives the local view $obs_a(\sigma)$ of every global event σ and runs its own monitor. Figure 3.1 illustrates this architecture for the positive case: the global stream is projected to each agent, processed by a local $(\mathcal{S}, \mathsf{P}e)$-knowledge monitor, and the individual verdicts are aggregated by disjunction.

Formally, for each agent $a \in Ag$ we consider an $(\mathcal{S}, \mathsf{P}e)$-knowledge monitor for a, and

we regard the decentralized diagnoser as the collection of these local monitors together with the rule that a diagnosis is produced as soon as at least one monitor accepts. This operational view corresponds to the disjunctive knowledge condition $\bigvee_{a \in Ag} \mathsf{K}_a \mathsf{P} e$ used in the definition of P-codiagnosability.

3.2 Bibliographic Notes

The notion of diagnosability for discrete-event systems was introduced by Sampath et al. [11], who defined what is now commonly referred to as F-diagnosability. In our logical framework, this notion corresponds to P-diagnosability and requires that the occurrence of an unobservable fault can be inferred after a bounded delay. The complementary perspective, sometimes called NF-diagnosability and focusing on the inference of fault absence, was studied later by Wang, Yoo, and Lafortune [13], who showed that the two notions are equivalent in the centralized setting.

Diagnosability in decentralized architectures was investigated by Debouk, Lafortune, and Teneketzis [6], leading to the notion of codiagnosability, where at least one agent must be able to make a correct diagnosis. In contrast to the centralized case, the equivalence between positive and negative diagnosability does not extend to decentralized settings, where positive and negative codiagnosability are incomparable.

Polynomial-time decision procedures for diagnosability and codiagnosability were established in the above works.

Bibliography

[1] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[2] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, 2011.

[3] Laura Bozzelli, Bastien Maubert, and Aniello Murano. The complexity of model checking knowledge and time. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1595–1601. ijcai.org, 2019.

[4] Jeremy W. Bryans, Maciej Koutny, Laurent Mazaré, and Peter Y. A. Ryan. Opacity generalised to transition systems. *Int. J. Inf. Sec.*, 7(6):421–435, 2008.

[5] Mika Cohen and Alessio Lomuscio. Non-elementary speed up for model checking synchronous perfect recall. In Helder Coelho, Rudi Studer, and Michael J. Wooldridge, editors, *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 1077–1078. IOS Press, 2010.

[6] Rami Debouk, Stéphane Lafortune, and Demosthenis Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discret. Event Dyn. Syst.*, 10(1-2):33–86, 2000.

[7] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[8] Bernd Finkbeiner and Julian Siber. Explainability requirements as hyperproperties. *Acta Informatica*, 62(4):38, 2025.

[9] Stéphane Lafortune, Feng Lin, and Christoforos N. Hadjicostis. On the history of diagnosability and opacity in discrete event systems. *Annu. Rev. Control.*, 45:257–266, 2018.

[10] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *J. Log. Algebraic Methods Program.*, 78(5):293–303, 2009.

[11] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control.*, 40(9):1555–1575, 1995.

[12] Ron van der Meyden and Nikolay V. Shilov. Model checking knowledge and time in systems with perfect recall (extended abstract). In C. Pandu Rangan, Venkatesh Raman, and Ramaswamy Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings*, volume 1738 of *Lecture Notes in Computer Science*, pages 432–445. Springer, 1999.

[13] Yin Wang, Tae-Sic Yoo, and Stéphane Lafortune. Diagnosis of discrete event systems using decentralized architectures. *Discret. Event Dyn. Syst.*, 17(2):233–263, 2007.