Selective Monitoring Without Delay for Probabilistic System

Julie Parreaux[1]
Supervisor: Stefan Kiefer[2]

[1] ENS Rennes
Julie.Parreaux@ens-rennes.fr
[2] Oxford University, UK
stekie@cs.ox.ac.uk

May 21, 2018 to August 17, 2018

Abstract.

Monitoring is an efficient tool to check correctness in runtime. A monitor observes the output produced by a Markov Chain and decides if the run is correct or faulty. A selective monitor chooses the observed letters with a policy. It skips letters to reduce the monitoring overhead. A selective monitor without delay preserves the time to make the decision.

We study selective monitoring without delay on labelled Markov Chains. We design a policy without delay. In the general case, this policy is not optimal, and we compute it in PSPACE. However, we introduce a particular class of MC: the non-Hidden Markov Chain. In this class, the policy is optimal, and we compute it in P.

Keywords: Runtime monitoring · Probabilistic system · Markov chain · Automata

1 Introduction

Fault detection is a crucial and challenging task in the automatic control of large complex systems. Several formal verification methods try to ensure the correctness of these systems. One of this methods is monitoring. It can check the correctness of systems at runtime. It is an alternative to testing and verification. Testing a system increases the confidence in the tested system but it cannot guarantee correctness. Verification can guarantee correctness but it is not feasible at runtime. Monitoring is a trade-off between these two methods. It is equivalent to real-time model checking in runtime. It is feasible and can guarantee the correctness of the current run.

In runtime verification, monitoring consists of deciding if the run is faulty. The system sends some messages to a monitor which decides. Efficient monitoring does not add lots of overhead to the execution of the system. It must inform on the faulty event as soon as possible. Else, the monitor is not useful. Overhead must be minimized. For some problems, such as the Quality Virtual Machine (QVM) problem [1] and the Adaptative Runtime Verification (ARV) problem [3], there exist efficient techniques to monitoring with a budget overhead. Frequently, these techniques use probability to reduce overhead.

We use probabilistic systems to make an efficient monitor. Probability can reduce the monitoring overhead. It introduces uncertainty in the analysis. The Runtime Verification with State Estimation (RVSE) problem [12] models some runtime gaps with probabilistic models. Sampling reduces the monitoring overhead. Probability reduces the overhead of the monitoring but it introduces uncertainty on the model and the result. The problem of Runtime Verification with Particle Filtering (RVPF) [9] studies the trade-off between the uncertainty and the overhead of the monitoring. Our approach is the opposite: we ask for the smallest overhead achievable without compromising precision at all.

Monitoring probabilistic systems is an important problem. We consider a Markov Chain with labelled transitions and a finite automaton which accepts a language of infinite words. Computing the probability that the random word emitted by this Markov Chain is accepted by this automaton is a classical problem. It is at the heart of probabilistic systems. In this work, we assume that a finite prefix already may determine if an infinite word is accepting. Compute this probability is a non-trivial *diagnosability* problem.

The diagnosability is the first problem when we would monitor a system. The existence of a monitor is a prerequisite for studying the monitor performance. The *diagnosability* problem asks if a monitor exists for a given system. For a class of models, two sub-questions solve this problem. First, some properties decide the existence of a monitor for the model. Model characteristics define these properties. Second, we build this monitor. This problem, initially, has been studied for discrete event systems with Labelled Transition System (LTS) [10]. It was shown that diagnosability can be decided in polynomial time but building monitor is in EXPTIME [8]. Then, this problem has been extended to the probabilistic systems with labelled Markov Chain [13]. Many notions of probabilistic systems diagnosability exist and have several names [11,4]. For all of these different notions, with only one exception, diagnosability is a PSPACE-complete problem [4,5].

Given a monitor, we can study its performance. Observations produce overhead in the monitoring. We would build a monitor to skip several observations. The monitor observes when it wants some information about the system. Selective monitoring designs a policy for the monitor. This policy decides when the monitor can skip observations. However, the probability of deciding if the run is correct or faulty may decrease when we skip an observation. In a faulty run, the policy might skip the unique “fault” letter. In this case, we can not know if the observation is faulty. We do not study this trade-off, we require that the policy is feasible (i.e. the probability of deciding does not decrease when we skip a letter). Our goal is to reduce monitoring overhead of a feasible optimal policy.

A policy is optimal when the expected cost of decision of this policy is minimal for all feasible policies. The cost of decision is the number of observations made during a run of the system. It was shown [6] that whether there is a feasible policy whose excepted cost is smaller than a given threshold is undecidable, even for diagnosable systems. We identify a class of Markov Chains, the non-Hidden Markov Chain. In this class, each label identifies a unique state. Non-Hidden Markov Chains are always diagnosable and there exists a feasible optimal policy [6].

The feasible optimal policy designed in [6] decides with a delay. A policy which observes all emitted letters decides before the optimal policy. Monitoring without delay is an important property. A useful monitor alerts if the run is faulty before the bad event arrives. A policy with delay may alert later. We would like to design an optimal feasible policy without delay. This policy decides at the same time as a policy which observes all letters. Moreover, this policy observes the fewest possible letters. We design this policy (algorithm 2) and prove that it is feasible and without delay in the general case (Theorem 2). We study the cost of decision of the policy and prove it not optimal in the general case (Theorem 3). For non-Hidden Markov Chains, we prove that this policy is optimal (Theorem 5). We can compute it in polynomial time (Corollary 3).

For the rest of the document, we give some notions needed to build our policy (section 2). We discuss more precisely optimal feasible policies (section 3). We present our feasible policy without delay in the general case (section 4). Then, we study properties of this policy on of the non-Hidden Markov Chain case (section 5).

2 Preliminaries

A (labelled) Markov Chain is a graph with weighted transitions that emitted one letter at each step. A right monitor to this is a Deterministic Finite Automaton. Its accepted language is the Markov Chain emitted word when the run is correct. An observation policy sends a letter or a skip symbol to the monitor. It choices when the monitor skips a letter. A belief describes the possible state in the Markov Chain and the Deterministic Finite Automaton after an observation. We use belief to study observations.

2.1 Markov Chain and Deterministic Finite Automaton

Definition 1. *A (discrete-time, finite-state, labelled)* Markov Chain (MC) *is a quadruple* (S, Σ, M, s_0) *where:*

- S *is a finite, nonempty, set of states;*
- Σ *is a finite, nonempty, alphabet;*
- s_0 *is an initial state;*
- $M : \Sigma \longrightarrow [0,1]^{S\times S}$ *specifies the transition such that* $\sum_{a\in\Sigma} M(a)$ *is a stochastic matrix, i.e. the value of each rows sums* 1.

Intuitively, if the MC is in the state s, it emits a and moves to s' with the probability $M(a)(s,s')$. Moreover, we can extend the definition of $M : \Sigma^* \longrightarrow [0,1]^{S\times S}$. If the MC is in the state s, it emits $u \in \Sigma^*$ and moves to s' with the probability $M(u)(s,s')$.

Example 1. In the MC in the Figure 3, $M(a)(s_0,s_1) = \frac{1}{4}$. When the MC is in s_0 it emits a and moves in s_1 with probability $\frac{1}{4}$. As the same, $M(bbb)(s_0,s_3) = \frac{1}{4}$.

We identify a class of MC: the non-Hidden Markov Chains. A non-Hidden Markov Chain identifies each state with a unique letter.

Definition 2. *An MC* $\mathcal{M} = (S, \Sigma, M, s_0)$ *is called* non-Hidden *if the emitted letter identifies the next state, i.e., there exists a function* $\overrightarrow{\cdot} : \Sigma \rightarrow S$ *such that* $M(a)(s,s') > 0$ *implies* $s' = \overrightarrow{a}$.

Example 2. In the non-Hidden MC (Figure 1), the state s_0 is identified by a, the state s_1 by e, the state s_2 by c and the state s_3 by b. In particular, if we observe the letter a, the next state of the MC must be s_0.

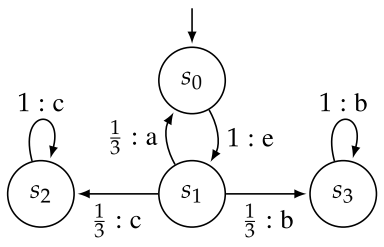

Fig. 1: An non-Hidden MC.

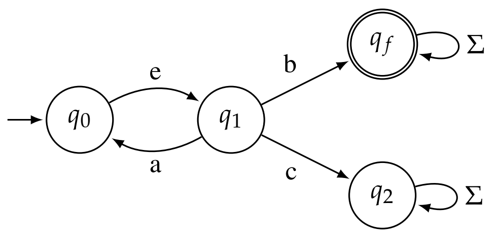

Fig. 2: DFA to monitor the non-Hidden MC.

Definition 3. *A* Deterministic Finite Automaton (DFA) *is a quintuple* $(Q, \Sigma, \delta, q_0, F)$ *where:*

- Q *is a finite, nonempty, set of states;*
- Σ *is a finite, nonempty, alphabet;*
- $\delta : Q \times \Sigma \longrightarrow Q$ *is a transition function;*
- q_0 *is an initial state;*
- $F \subseteq Q$ *is a set of accepting states.*

A DFA defines a language over infinite words $L \subseteq \Sigma^\omega$ *as follows:*

$$L = \{w \in \Sigma^\omega \mid \delta(q_0, u) \in F \text{ for some prefix } u \text{ of } w\}$$

This definition does not require accepting states are visited infinitely often: only one time suffices. Moreover, we can assume, without lost generality, there is a unique accepting state ($F = \{f\}$) and $\delta(f, a) = f$ for all $a \in \Sigma$. Else, we build a new state as the unique accepting state. For each accepting state and all letter of Σ, we add a transition between it and the new state. These states are not accepting states in the DFA.

Example 3. The DFA (Figure 2) defines the language $L = e(ae)^*b\Sigma^w$.

We can extend the definition of a DFA. We introduce nondeterminism in it to obtain a *Nondeterminism Finite Automaton (NFA)*. An NFA is a quintuple $(Q, \Sigma, \delta, Q_0, F)$, as a DFA, where $Q_0 \subseteq Q$ is a nonempty set of initial states and $\delta : Q \times \Sigma \to 2^Q$ is the transition function. The NFA defines the language $L = \{w \in \Sigma^\omega \mid \delta(q_0, u) \in F$ for some prefix u of $w\}$.

For the rest of the document, we fix the notations of an MC $\mathcal{M} = (S, \Sigma, M, s_0)$ and a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$.

2.2 Observation policy

An observation policy over an MC and its DFA is a choice between observing or skipping the next emitted letter. An observation is the result of one of these choices. It is either a letter of Σ or a special symbol $\perp \notin \Sigma$. It is defined on a new alphabet $\Sigma_\perp = \Sigma \cup \{\perp\}$.

Definition 4. *An* observation policy $\rho : \Sigma_\perp^* \longrightarrow \{0,1\}$ *is a function that, given the observation made so far, says whether we should observe the next letter.*

For each observation policy ρ, we can associate a *projection* $\pi_\rho : \Sigma^\omega \longrightarrow \Sigma_\perp^\omega$ such that $\pi_\rho(a_1 a_2 ...) = o_1 o_2 ...$ where

$$o_{n+1} = \begin{cases} a_{n+1} \text{ if } \rho(o_1...o_n) = 1 \\ \perp \text{ if } \rho(o_1...o_n) = 0 \end{cases} \text{for all } n \geq 0$$

The *see-all policy*, denoted by $\bullet$, is such that $\pi_\bullet(w) = w$, i.e., it does not skip any letter.

Two observations are in relation $o_1 \sim o_2$ if and only if o_1 and o_2 are the same letter in Σ or at least of them is equal to $\perp$. We extend this relation for the (infinite or finite) sequence of the same length. We denote $w \gtrsim v$ when $u \sim v$ holds for the length-$|v|$ prefix of w.

Example 4. The DFA in the Figure 4 denotes the language $L = \{u\}b\Sigma^w$ with $u \in \Sigma^2$. For all $w \in L$, we have $w \gtrsim \perp\perp b$. For all length-$|3|$ prefix of $w \in L$, u, $u \sim \perp\perp b$. We have $u = u'b$ with $u' \in \Sigma^2$ and $u' \sim \perp\perp$.

An observation can be infinite. Only a finite prefix of observation is needed to characterise this one. We distinguish two types of prefix: prefixes emit by the MC (called enabled) and the other ones. The rest of prefix type is defined in function of what happens if we observe all, now. All infinite completed word of a negatively deciding prefix describes a faulty run. Reciprocally, all infinite completed word of a positively deciding prefix describes a correct run. If a prefix is positively or negatively deciding, it is a *deciding* prefix. An infinite word is deciding when one of these prefixes is deciding. A confused prefix can be complete in a not deciding word. A very confused cannot be complete in a deciding word. Finally, a prefix is said finitary if it is deciding or very confused.

Definition 5. *For all observation prefix v:*

- v *is* enabled *when* $Pr(\{w \gtrsim v\}) > 0$*;*
- v *is* negatively deciding *when* $Pr(\{w \gtrsim v \mid w \in L\}) = 0$*;*
- v *is* positively deciding *when* $Pr(\{w \gtrsim v \mid w \notin L\}) = 0$*;*
- v *is* confused *when* $Pr(\{vu \mid vu \textit{ deciding}\}) < Pr(\{uw \mid u \sim v\})$*;*
- v *is* very confused *when* $Pr(\{vu \mid vu \textit{ deciding}\}) = 0$*;*
- v *is* finitary *when* $Pr(\{vu \mid vu \textit{ deciding or very deciding}\}) = Pr(\{uw \mid u \sim v\})$.

Example 5. For the non-Hidden MC (Figure 1) and its DFA (Figure 2), $\perp\perp\perp b$ is a positively deciding prefix. For all w which complete $\perp\perp\perp b$, $\perp\perp\perp bw \in L$. Reciprocally, $\perp\perp\perp c$ is a negatively deciding prefix. There does not exist any way to complete $\perp\perp\perp c$ such that this word is in L. When $w \in L$, the third letter of w is b.

Definition 6. *An observation policy ρ* decides *w if and only if $\pi_\rho(w)$ has a deciding prefix.*

Lemma 1. (cf. [6, lemma 1]) *For any w, if some policy decides w then $\bullet$ decides w.*

The lemma 1 implies $max_\rho(Pr(\{w \mid \rho \text{ decides } w\})) = Pr(\{w \mid \bullet \text{ decides } w\})$. A feasible policy reaches this maximum.

Definition 7. *A policy ρ is a* feasible policy *if and only if $Pr(\{w \mid \rho \textit{ decides } w\}) = Pr(\{w \mid \bullet \textit{ decides } w\})$, whether, $Pr(\{w \mid \bullet \textit{ decides } w \textit{ implies } \rho \textit{ decides } w\}) = 1$.*

A *monitor* is an algorithm to implement a policy. For each letter (in the entrance), it returns “yes”, “no” or $n \in \mathbb{N}$ where n is the number of skipped letters.

2.3 Belief

A *belief* is a subset of states from the product $S \times \mathcal{Q}$ where it could be now. For each observation v, the function $\Delta(B, v)$ updates the belief B. The result of this function is the set of the reachable states from B with v. A *belief NFA* $\mathcal{B}$ is the NFA based on the composition between an MC and a DFA. We have $\mathcal{B} = (S \times \mathcal{Q}, \Sigma_\perp, \Delta, B_0, \emptyset)$ where $B_0 = \{(s_0, q_0)\}$ and:

$$\Delta((s,q),a) = \{(s',q') \mid M(a)(s,s') > 0,\ \delta(q,a) = q'\} \text{ for } a \in \Sigma$$
$$\Delta((s,q),\perp) = \bigcup_{a\in\Sigma} \Delta((s,q),a)$$

Reasoning algorithmically on belief is easier than on prefixes. We define the relation between prefixes and beliefs. For any $s \in S$, we denote Pr_s the probability measure of the MC $\mathcal{M}_s$ which is obtained from $\mathcal{M}$ when s is the initial state. As the same, for any $q \in \mathcal{Q}$, we denote $L_q \subseteq \Sigma^\omega$ the language of the DFA $\mathcal{A}_q$ which is obtained from $\mathcal{A}$ when q is the initial state.

Definition 8. *For any pair (s,q) we define:*

- *(s,q) is* negatively deciding *if and only if $Pr_s(L_q) = 0$;*
- *(s,q) is* positively deciding *if and only if $Pr_s(L_q) = 1$.*

We extend the definition to the belief.

Definition 9. *For all belief B, B is:*

- negatively (positively) deciding *if and only if all its elements are;*
- confused *if and only if $Pr_s(\{uv \mid \Delta(B,u) \textit{ is deciding}\}) < 1$ for some $(s,q) \in B$;*
- very confused *if and only if $\Delta(B,u)$ is empty or not deciding for all u;*
- finitary *if and only if $Pr_s(\{uv \mid \Delta(B,u) \textit{ is deciding or very confused}\}) = 1$ for all $(s,q) \in B$.*

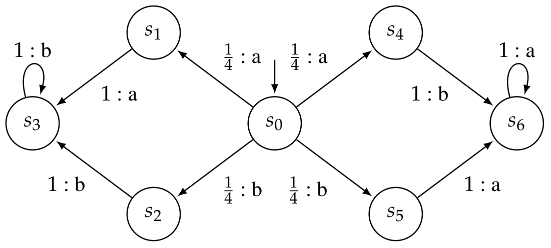

Fig. 3: A general MC.

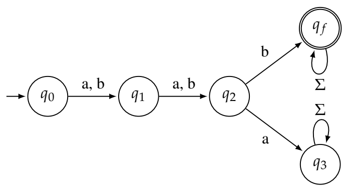

Fig. 4: DFA to control the previous general MC.

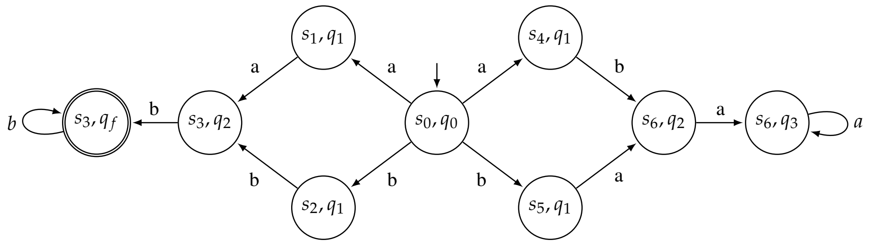

Fig. 5: A natural representation of belief pairs from the MC (Figure 3) and its DFA (Figure 4). We have represented only non-$\perp$ observations.

Example 6. Consider the MC in Figure 3 and its DFA (Figure 4). We fix $B_0 = \{(s_0, q_0)\}$. A natural representation of belief pairs from this MC and its DFA is given in Figure 5.

$\Delta(B_0, a) = \{(s_1, q_1), (s_4, q_1)\}$; $\Delta(B_0, \perp) = \{(s_1, q_1), (s_4, q_1), (s_2, q_1), (s_5, q_1)\}$
$\Delta(B_0, aa) = \{(s_3, q_2)\}$; $\Delta(B_0, \perp a) = \{(s_3, q_2), (s_6, q_2)\}$

The belief $\Delta(B_0, aa)$ is (positively) deciding because (s_3, q_2) is positively deciding: $Pr_{s_3}(L_{q_2}) = Pr_{s_3}(b^\omega) = 1$. However, $\Delta(B_0, \perp a)$ is not deciding. Indeed, (s_3, q_2) is positively deciding and (s_6, q_2) is negatively deciding ($Pr_{s_6}(L_{q_2}) = Pr_{s_6}(b^\omega) = 0$). It is also not confused because (s_3, q_2) and (s_6, q_2) are deciding.

Lemma 2. *(*cf. [6, lemma 4]*) Let v be an observation prefix.*

1. *v is enabled if and only if $\Delta(B_0, v) \neq \emptyset$.*
2. *v is negatively deciding if and only if $\Delta(B_0, v)$ is negatively deciding.*
3. *v is positively deciding if and only if $\Delta(B_0, v)$ is positively deciding.*
4. *v is confused if and only if $\Delta(B_0, v)$ is confused.*
5. *v is very confused if and only if $\Delta(B_0, v)$ is very confused.*
6. *v is finitary if and only if $\Delta(B_0, v)$ is finitary.*

Lemma 3. *For all belief B and $v \in \Sigma^*$, $\Delta(B, \epsilon)$ is deciding $\Rightarrow \Delta(B, v)$ is deciding.*

We fix, for the rest of this document, these notations: for all $a \in \Sigma_\perp$, $B_a = \Delta(B, a)$; $B_{\rho,i}$ is the belief computed by the policy ρ in step i and $B_{\rho,i,a} = \Delta(B_{\rho,i}, a)$. When we have no doubt about the policy ρ, we can suppress ρ of the notation to avoid clutter.

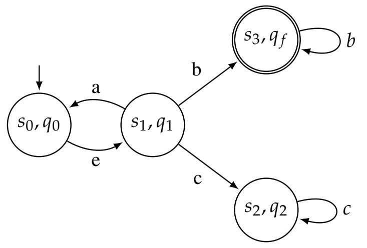

Fig. 6: A natural representation of belief pairs from the non-Hidden MC (Figure 1) and its DFA (Figure 2). We have represented only non-$\perp$ observations.

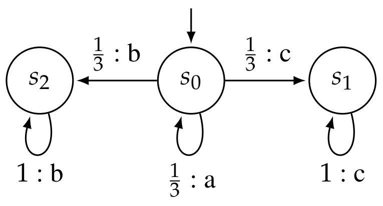

Fig. 7: An MC with an arbitrarily long length shortest deciding prefix.

For all belief B, we define the language $L_B = L_B^d \cap L_B^e$ with $L_B^d = \{v \mid \Delta(B,v) \text{ is deciding}\}$ and $L_B^e = \{v \in \Sigma^* \mid \Delta(B,v) \neq \emptyset\}$. Intuitively, the language L_B is the language of the enabled deciding prefix from B.

Example 7. In the representation of belief pairs in Figure 6, let $B = \{(s_0, q_0)\}$. We have $L_B^d = \Sigma^* \setminus \{(ea)^*\}$ and $L_B^e = e(ae)^*\{b+c\}\Sigma^*$. So, $L_B = e(ae)^*\{b+c\}\Sigma^*$.

Lemma 4. *For all belief B.*

1. $L_B \subseteq \Sigma^*$ *and* $L_B \supsetneq \Sigma^*$.
2. $v \in L_B \Leftrightarrow \Delta(B,v)$ *is deciding and* $\Delta(B,v) \neq \emptyset$.
3. *For all* $a \in \Sigma$, $L_{B_a}^e \subseteq L_{B_\perp}^e$.
4. $L_{B_\perp} \subseteq \bigcup_{a \in \Sigma} L_{B_a}$.

2.4 Cost of decision

The cost of decision for a policy ρ and a word w is noted by $C_\rho(w)$. It is the number of letters observed by the policy to decide w.

Definition 10. *For all policy ρ and word w, let $v = o_1 o_2 ...$ be the shortest deciding prefix of $\pi_\rho(w)$. The* cost of decision *for ρ and w is defined as follows:*

$$C_\rho(w) = \sum_{k=0}^{|v|-1} p(o_1 ... o_k)$$

Our goal is to design a feasible optimal policy regarding the cost of decision. We note $Ex(C_\rho)$ the expectation of the cost of decision of this policy ρ. The expectation respects to the probability distribution induced by the MC. We would find a feasible policy without delay that minimizes $Ex(C_\rho)$. Finding a feasible policy that minimizes the expectation of the cost of decision is equivalent to compute for all feasible policy ρ without delay:

$$c_{inf} = \inf_\rho Ex(C_\rho)$$

3 An optimal policy in the general case

The *lighted see-all* policy is a naive policy to decide if a given run is correct. It observes all letters until the observation prefix is finitary. However, observe a letter has an expensive cost. Our goal is to minimize this cost. We would a feasible policy to decide with a minimum number of observations. To simplify the problem, we suppose the considered system has a diagnoser. We can check is it on PSPACE [4, Theorem 8].

Define a feasible policy with an optimal expectation of the number of observing letter is an optimization problem. The decision problem associated with this one asks if there is a feasible policy whose expected cost is smaller than a given threshold. It is undecidable in the general case [6, Theorem 15].

In the non-Hidden MC case, the procrastination policy is an optimal feasible policy [6]. The procrastination is the maximal number of skipping letters such that the obtained belief is not confused. In other words, the procrastination number is the maximal of letters such that the policy can skip since it becomes not feasible. The procrastination number definition is based on : for a belief B and $k \in \mathbb{N}$, if $\Delta(B, \perp^k)$ is confused then $\Delta(B, \perp^{k+1})$ is also confused.

Definition 11. *For all belief B, the* procrastination number *is:*

$$cras(B) = sup\{k \in \mathbb{N} \mid \Delta(B, \perp^k) \textit{ is not confused}\} \in \mathbb{N} \cup \{-1, \infty\}$$

If B is confused $cras(B) = -1$.

Example 8. For the belief $B = \{(s_0, q_0)\}$ in Figure 6, we have $cras(B) = \infty$. Indeed, $\forall k \in \mathbb{N}$, $\Delta(B_0, \perp^k)$ is not confused.

The procrastination policy (Algorithm 1) skips the procrastination number letters. This number can be equal to the infinity. A policy cannot skip an infinite number of letters. The procrastination policy is parametrized by a larger $K \in \mathbb{N}$. We define $\rho_{pro}(K)$ with the monitor define by the Algorithm 1 and, for all belief B, $k(B) = min(K, cras(B))$.

Algorithm 1 Procrastination policy $\rho_{pro}(K)$ [6]

```
1: $i \leftarrow 0$; $B_0 = \{s_0, q_0\}$
2: **while** $B_i$ is not deciding **do**
3:     skip $k(B_i)$ observations, then observe a letter $a_i$
4:     $B_{i+1} \leftarrow \Delta(B_i, \perp^{k(B_i)} a_i)$
5:     $i \leftarrow i + 1$
6: **end while**
7: Output yes/no decision
```

Example 9. Apply this policy on the word $w = aab^\omega$ in the MC in Figure 3 and its DFA (Figure 4). By [6, Theorem 27], we choose $K = |S|^2 * |Q|^2 + 1 = 7^2 * 5^2 + 1 = 1226$. As B_0 is not deciding and $cras(B_0) = \infty$ the policy skips the first K letters and observes

$a_0 = b$. We have $B_1 = \Delta(B_0, \perp^K b) = \{(s_3, q_f)\}$ and B_1 is deciding, the policy returns yes and $\pi_{\rho_{pros}}(w) = \perp^{1226} b$.

Theorem 1. *(*cf. [6, Lemma 23 and Theorem 27]*) For all $K \in \mathbb{N}$, the policy $\rho_{pros}(K)$ is a diagnoser. There exists K such that it is an optimal policy.*

We choose a value for K to apply the policy ρ_{pros}. The optimality of the policy is guaranteed if K is great. In the example 9, we need to choose $K = 1226$. The policy decides with the 1227th emitted letter. We can decide if we observe the two first letters. The policy ρ_{pros} decides with delay. This delay depends on the choice of the value of K. However, decide with a delay can be prohibited for some systems. In these systems, the policy must alert since the bad event will arrive. Our goal is to design a feasible optimal policy without delay.

4 A policy without delay

We want to a feasible optimal (regarding the cost of the decision) policy. It decides without delay. In other words, we would a policy that when the policy • decides, it can decide too . Our policy decides with a minimal number of observed letter. The constraint "without delay" is a strong constraint. A policy without delay skips an emitted letter when this letter gives any information about the run. It needs to guarantee that the shortest observed deciding prefix is always deciding when it skips the next letter.

Algorithm 2 Policy $\rho_{ZeroDelay}$

Main algorithm

```
1:  $i \leftarrow 0$; $B_0 \leftarrow (s_0, q_0)$
2:  while $B_i$ is not deciding do
3:      if can skip then
4:          $a_i \leftarrow \perp$
5:      else
6:          $a_i \in \Sigma$
7:      end if
8:      $B_{i+1} \leftarrow \Delta(B_i, a_i)$
9:      $i \leftarrow i + 1$
10: end while
11: Output yes/no decision
```

Compute if the policy can skip the next letter

```
1: Compute $B_\perp = \Delta(B_i, \perp)$
2: for all $a \in \Sigma$ do
3:     Compute $B_a = \Delta(B_i, a)$
4:     if $\exists w : \Delta(B_a, w) \neq \emptyset \wedge \Delta(B_a, w)$ is deciding $\wedge$
       $\Delta(B_\perp, w)$ is not deciding then
5:         Return False
6:     end if
7: end for
8: Return True
```

By definition, w is infinite: $w \in \Sigma^\omega$. In the practice, if B_a is deciding, $w = \epsilon$. The policy skips only it has already decided. We check the condition for $w \in \Sigma^*$. If $w \in \Sigma^*$ holds condition, $wu \in \Sigma^\omega$ holds too. The condition is checked only on the prefix.

We can build an MC such that w has the shortest deciding prefix with a length arbitrary long. For example, in the MC in Figure 7, for all $n \in \mathbb{N}$, there exist $w = a^n b^\omega$

such that $a^n b$ is the shortest deciding prefix. We test an infinite set of deciding prefix to check the policy skipping condition.

Example 10. Apply the policy $\rho_{ZeroDelay}$ (Algorithm 2) on the word $w = aab^\omega$ emitted by the MC in Figure 3 and its DFA (Figure 4).

B_0 is not deciding. We compute if the policy can skip the next letter. We have $B_{0,\perp} = \{(s_1, q_1), (s_2, q_1), (s_4, q_1), (s_5, q_1)\}$ and $B_{0,a} = \{(s_1, q_1), (s_4, q_1)\}$. Moreover, $\Delta(B_{0,a}, a) = \{(s_3, q_2)\}$ is deciding but $\Delta(B_{0,\perp}, a) = \{(s_3, q_2), (s_6, q_2)\}$ is not deciding (example 6). So, we need to observe the next letter: $a_0 = a$ and $B_1 = \{(s_1, q_1), (s_4, q_1)\}$.

B_1 is also not deciding. We compute if the policy can skip the next letter. We have $B_{1,\perp} = \{(s_3, q_2), (s_6, q_2)\}$ and $B_{1,a} = \{(s_3, q_2)\}$. As, $\Delta(B_{1,a}, \epsilon) = B_{1,a}$ is deciding and $\Delta(B_{1,\perp}, \epsilon) = B_{1,\perp}$ is not deciding (example 6). So, we need to observe the next letter: $a_0 = a$ and $B_2 = \{(s_3, q_2)\}$.

B_2 is deciding. Hence, w is positively deciding and $\pi_{\rho_{ZeroDelay}}(w) = aa$.

Our policy decides to skip the next letter with this condition: $(\forall v,\ \Delta(B_a, v) = \emptyset \vee (\Delta(B_a, v)$ is deciding $\Rightarrow \Delta(B_\perp, v)$ is deciding$))$. We cannot cut down the condition and guarantee the obtained policy is optimal and without delay. We cannot remove the condition ($\Delta(B_a, v)$ is deciding $\Rightarrow \Delta(B_\perp, v)$ is deciding). It guarantees that our policy is without delay. If it skips a letter, then the observed prefix is not decided without delay.

We cannot remove the condition $(\Delta(B_a, v) = \emptyset)$. It guarantees that our policy is optimal. Verify $(\Delta(B_a, v) = \emptyset)$ is easier than verify ($\Delta(B_a, v)$ is deciding $\Rightarrow \Delta(B_\perp, v))$ is deciding. By definition, $(\Delta(B_a, v) = \emptyset)$ implies $(\Delta(B_a, v))$. If there exists $a \in \Sigma$ such that $B_a = \emptyset$, the policy can skip the next letter if and only if $B_\perp$ is deciding. For example, let the MC in Figure 1 and its DFA (Figure 2). With our policy $\rho_{ZeroDelay}$, as $B_\perp = B_e$ and for $a \in \Sigma \setminus \{e\}$, $B_a = \emptyset$, we skip the first letter. The policy without this condition cannot skip the first letter. Indeed, $(B_a = \emptyset)$ is deciding but $(B_\perp = \{(s_1, q_1)\})$ is not deciding. Hence, without this condition, the policy is not optimal.

Feasible policy without delay Our policy $\rho_{ZeroDelay}$ is feasible and without delay policy. If v is a deciding prefix, the observation of v by $\rho_{ZeroDelay}$ is also deciding.

Theorem 2 **($\rho_{ZeroDelay}$ is feasible and without delays).** *For all deciding prefix v, $\pi_{\rho_{ZeroDelay}}(v)$ is also deciding.*

This theorem proves that $\rho_{ZeroDelay}$ is feasible because of $v = \pi_\bullet(v)$. It also proves that $\rho_{ZeroDelay}$ is without delay. If v is the shortest deciding prefix for v, $\rho_{ZeroDelay}$ decides v.

Idea of proof. Let v a deciding prefix. We prove by induction on the number of the policy application. The observed prefix build by $\rho_{ZeroDelay}$ at this step completed by the rest of v is also deciding.

Optimal policy We require that $\rho_{ZeroDelay}$ observes a minimal number of letters to decide. Our policy has to minimize $Ex(C_{\rho_{ZeroDelay}})$. However, we prove its non-optimality, in the general case.

Theorem 3 **($\rho_{ZeroDelay}$ is not optimal).** *For all feasible policy without delays ρ, we do not have necessarily: $Ex(C_{\rho_{ZeroDelay}}) \leq Ex(C_\rho)$.*

Idea of proof. We build a counterexample. The policy $\rho_{ZeroDelay}$ is not optimal on the MC in Figure 12 and its DFA (Figure 13). The MC probability distribution decides the next state (and the next emitted letter) and influences on the optimal policy choices.

Cost of decision The cost of decision of a policy is the number of policy observations to decide. It depends on the MC probability distribution and the current belief. We compute its expectation with a local approach: $Ex(C_{\rho_{ZeroDelay}}) = c(s_0, q_0)$. $c(s,q)$ is the expected cost of decision under $\rho_{ZeroDelay}$ from (s,q). We define $c(s,q)$ as follows:

$$c(s,q) = \begin{cases} 0 & \text{if } (s,q) \text{ is deciding} \\ \sum_{(s',q')} \sum_{a \in \Sigma} M_{\rho_{ZeroDelay}}(a)(skip + c(s',q')) & \text{otherwise} \end{cases}$$

where $skip = 1$ if and only if the policy does not skip the next letter and 0 otherwise.

The composition between an MC and a DFA induces the MC $\mathcal{M}_{\rho_{ZeroDelay}}$. Its alphabet contains a new symbol \$. This symbol marks when the policy decides. $c(s,q)$ labels the state (s,q) in $\mathcal{M}_{\rho_{ZeroDelay}}$ (see $\mathcal{M}_{pro}(K)$ in [6] for more details).

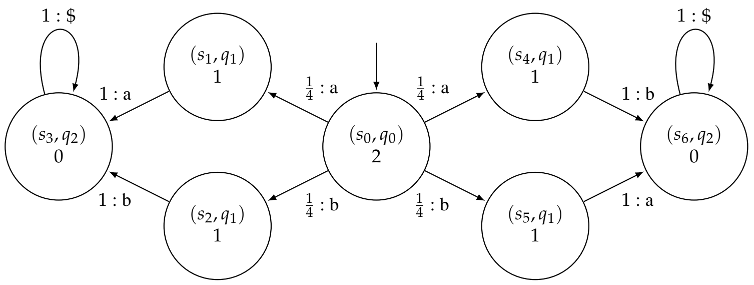

Fig. 8: The MC $M_{\rho_{ZeroDelay}}$ from the MC in Figure 3.

Example 11. We compute $Ex(C_{\rho_{ZeroDelay}})$ for the MC in Figure 3. We need to compute $c(s_0, q_0)$. In this MC, the policy does not skip any letter until it decides. So, $skip = 1$ for each (s,q). As (s_3, q_f) and (s_5, q_2) are deciding, $c(s_3, q_f) = c(s_5, q_2) = 0$. For each $s \in S$, $c(s, q_1) = 1$. Hence, $c(s_0, q_0) = 2\frac{1}{4} + 2\frac{1}{4} + 2\frac{1}{4} + 2\frac{1}{4} = 2$. This MC gives the $M_{\rho_{ZeroDelay}}$ in Figure 8.

Complexity We evaluate the complexity of the policy $\rho_{ZeroDelay}$. Compute the skip condition is the most difficult work in our policy evaluation. We prove that the policy evaluation is in PSPACE.

Theorem 4. *For all belief B and $a \in \Sigma$, we have:*

$$L_{B_a} \subseteq L_{B_\perp} \Leftrightarrow (\forall v,\ \Delta(B_a, v) = \emptyset \vee (\Delta(B_a, v) \textit{ is deciding} \Rightarrow \Delta(B_\perp, v) \textit{ is deciding}))$$

Idea of proof. Let B be a belief and $a \in \Sigma$. We apply the definition of L_B and its properties in the lemma 4.

Corollary 1. *Given an MC $\mathcal{M}$ and a DFA $\mathcal{A}$, the $\rho_{ZeroDelay}$ evaluation is in PSPACE.*

Proof. For all $a \in \Sigma$, we decide the skipping condition to compute $L_{B_a} \subseteq L_{B_\perp}$, by the Theorem 4. The inclusion of NFA languages is a problem PSPACE-complete [7]. Hence, evaluate the policy is a problem PSPACE.

5 Non-Hidden Markov Chain case

We restrict our study to the non-Hidden MC case. In this section, we consider only the non-Hidden MC. For this class, each label identifies only one state. The policy $\rho_{ZeroDelay}$ is optimal and we can compute it in P.

Optimal policy In general, the policy $\rho_{ZeroDelay}$ is not optimal. For a non-Hidden MC, we prove it is an optimal policy. In this MC's class, our policy minimizes $Ex(C_{\rho_{ZeroDelay}})$.

Theorem 5 **($\rho_{ZeroDelay}$ is optimal in the non-Hidden MC case).** *Given a non-Hidden MC and a DFA. For all feasible policy without delays ρ, we have $Ex(C_{\rho_{ZeroDelay}}) \leq Ex(C_\rho)$.*

Idea of proof. Let ρ a feasible policy without delay. We prove by induction on the number of applications of the policies $\rho_{ZeroDelay}$ and ρ. The policy $\rho_{ZeroDelay}$ observes less letters than ρ or $\rho_{ZeroDelay}$. If ρ and $\rho_{ZeroDelay}$ observe the same number of letters, the language from $B_{\rho_{ZeroDelay},n}$ is included in the language from $B_{\rho,n}$.

Cost of decision The optimally result assumes that $c_{inf} = Ex(C_{\rho_{ZeroDelay}})$. By the study of the cost of decision, $c_{inf} = c(s_0, q_0)$.

Example 12. We compute c_{inf} for the non-Hidden MC (Figure 1). By the previous remark, we need compute $Ex(C_{\rho_{ZeroDelay}}) = c(s_0, q_0)$. In this MC, the policy observes only in the state (s_1, q_1). As (s_2, q_f) and (s_3, q_2) are deciding, $c(s_2, q_f) = c(s_3, q_2) = 0$. We have $c(s_1, q_1) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3}(1 + c(s_0, q_0)) = 1 + \frac{1}{3}c(s_0, q_0)$. Hence, $c(s_0, q_0) = \frac{3}{2}$ and $c_{inf} = \frac{3}{2}$. This MC gives the $M_{\rho_{ZeroDelay}}$ in Figure 9.

We compute c_{inf} in P. Compute c_{inf} is equivalent to compute $c(s_0, q_0)$. $c(s_0, q_0)$ satisfies a system of linear equation where coefficients are come from the MC. We solve this system in P. Moreover, the policy decides if it skips the next letter in P (see Corollary 3).

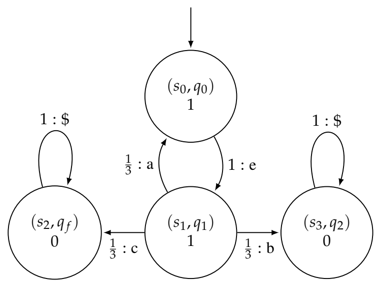

Fig. 9: $M_{\rho_{ZeroDelay}}$ from the MC in Figure 1.

Complexity In the general case, we compute $\rho_{ZeroDelay}$ in PSPACE. However, in the non-Hidden MC case, we can compute $\rho_{ZeroDelay}$ in P. If ϵ holds the skipping condition of $\rho_{ZeroDelay}$ then all word w the condition holds.

Theorem 6. *For all belief B such that $B_\perp$ is not confused and for all $a \in \Sigma$,*

$$L_{B_a} \subseteq L_{B_\perp} \Leftrightarrow (\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$$

Idea of proof. Let B a belief such that $B_\perp$ is not confused and $a \in \Sigma$. We use some properties from the language equivalence theory (section C). The non-Hidden MC structure gives these properties. We use the definition of L_B and the lemma 4.

Corollary 2. *For all belief B and $a \in \Sigma$,*

$$(B_a = \emptyset \vee B_a \text{ is deciding } \Rightarrow B_\perp \text{ is deciding}) \Leftrightarrow (\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$$

Example 13. Apply the police with this new condition on the word $w = eaeb^\omega$ emitted by the non-Hidden MC (Figure 1) and its DFA (Figure 2). The skipping condition is proved in the Corollary 2.

B_0 is not deciding. We have $B_e = \{s_1, q_1\}$ is not deciding and for all $a \in \Sigma \setminus \{e\}$, $B_a = \emptyset$. We have, for all $a \in \Sigma$, $(B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding$)$. We skip the next letter and $a_0 = \perp$ and $B_1 = \{(s_1, q_1)\}$.

B_1 is not deciding. $B_b = \{s_3, q_2\}$ is deciding and $B_\perp = \{(s_0, q_0), (s_2, q_2), (s_3, q_f)\}$ is not deciding. B_b deciding does not imply $B_\perp$ deciding. We observe the next letter and $a_1 = a$ and $B_2 = \{(s_0, q_0)\}$.

As $B_2 = B_0$, we use the same reasoning of B_0. We skip the next letter and $a_2 = \perp$ and $B_3 = \{(s_1, q_1)\} = B_1$. As the same $B_3 = B_1$, we observe the next letter $a_3 = b$ and $B_4 = \{(s_3, q_f)\}$.

B_4 is deciding and $\pi_{\rho_{ZeroDelay}}(w) = \perp a \perp b$. This policy is more efficient than the $\bullet$ policy. Its cost of decision is less than the cost of decision of $\bullet$.

Compute $(B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding$)$ is in P [6, lemma 5]. In the general case, check if $B_\perp$ is confused, is in PSPACE [6, Lemma 5]. However, the equivalent language in the for non-Hidden MC case gives a polynomial time algorithm to check it (section C).

Corollary 3. *Given a non-Hidden MC $\mathcal{M}$ and a DFA $\mathcal{A}$, we compute $\rho_{ZeroDelay}$ in P.*

Proof. Let a non-Hidden MC $\mathcal{M}$ and a DFA $\mathcal{A}$. The skipping condition of $\rho_{ZeroDelay}$ is equivalent to $(B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding$)$ if $B_\perp$ is not confused (Theorem 6 and Corollary 2). In the non-Hidden MC case, check if $B_\perp$ is not confused is in P (see section C). Moreover, compute $(B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding$)$ if $B_\perp$ is in P [6, lemma 5]. Hence, we compute $\rho_{ZeroDelay}$ in P.

6 Conclusion

We want to design a feasible optimal policy without delay regarding the cost of decision. We have designed a feasible policy without delay $\rho_{ZeroDelay}$ (Algorithm 2). For each letter, it choices between observing or skipping the next letter. It skips the next letter if and only if this one gives any information to decide. The skipping condition checks the inclusion between two NFAs languages (Theorem 4). We compute this policy in PSPACE (Corollary 1). We have also proved the non-optimality of $\rho_{ZeroDelay}$ (Theorem 3). The optimal policy choices depend on the MC probability distribution.

We have introduced a particular class of MC: the non-Hidden MC. In this class, each letter of the alphabet identifies a unique state. For a non-Hidden MC, $\rho_{ZeroDelay}$ is optimal (Theorem 5). Moreover, we can compute this policy in P (Corollary 3).

We have started to introduce a new class of MC: Deterministic MC (section B). In this class, each state is identified with a letter and a state. It is between a non-Hidden MC and a general MC. We hope to obtain better results than in the general case. However, we prove the non-optimality of $\rho_{ZeroDelay}$ for a Deterministic MC (Theorem 7). We use the same counterexample of the general case. Moreover, we have not any gain on the complexity in this case. We compute $\rho_{ZeroDelay}$ in PSPACE (Theorem 8).

In the general case and for the Deterministic MC, we will prove a hardness result on the complexity to compute a feasible optimal policy without delay. We will prove that an optimal choice between observing or skipping the first letter is PSPACE-hard (or at least NP-hard). The MC probability distribution influences the optimal choice between observing or skipping a letter. We will prove the feasible optimal policy without delay is undecidable, in the general case. In the Deterministic MC, we hope to obtain a better result: the feasible optimal policy without delay is decidable.

We can test our policy $\rho_{ZeroDelay}$ in the practice. We will implement the policy and observe the result about its cost of decision on real systems. We will study if the selective monitoring without delay is efficient. In our work, we use only feasible policy. We will study the trade-off between precision (probability to decide) and overhead (number of observation). To finish, we will study $\rho_{ZeroDelay}$ in different classes of MC.

Acknowledgement

I wish to thank all the people who allowed me to do my internship in the best possible conditions.
I wish to thank especially my supervisor for his welcome, his time and his help throughout this internship. Thanks to him, I was able to attend FLoC and it was a great experience for me. This has allowed me to confirm my interest in research.

I also wanted to thank the people with whom I shared my office for their welcome and help, among other things, to discover Oxford University.
I will end with a special thank you for Nathalie Bertrand who gave me the opportunity to do this internship. His help and support were of great importance to me.

References

1. Matthew Arnold, Martin Vechev, and Eran Yahav. Qvm: An efficient runtime for detecting defects in deployed systems. *SIGPLAN Not.*, 43(10):143–162, October 2008.
2. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.
3. Ezio Bartocci, Radu Grosu, Atul Karmarkar, Scott A. Smolka, Scott D. Stoller, Erez Zadok, and Justin Seyster. Adaptive runtime verification. In Shaz Qadeer and Serdar Tasiran, editors, *Runtime Verification*, pages 168–182, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
4. Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Foundation of Diagnosis and Predictability in Probabilistic Systems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'14)*, New Delhi, India, December 2014.
5. Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Accurate approximate diagnosability of stochastic systems. In Adrian-Horia Dediu, Jan Janoušek, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications*, pages 549–561, Cham, 2016. Springer International Publishing.
6. Radu Grigore and Stefan Kiefer. Selective monitoring. *CoRR*, abs/1806.06143, 2018.
7. John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.
8. Shengbing Jiang, Zhongdong Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, Aug 2001.
9. Kenan Kalajdzic, Ezio Bartocci, Scott A. Smolka, Scott D. Stoller, and Radu Grosu. Runtime verification with particle filtering. In Axel Legay and Saddek Bensalem, editors, *Runtime Verification*, pages 149–166, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
10. Meera Sampath, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen, and D Teneketzis. Diagnosability of discrete event systems. 40:1555 – 1575, 10 1995.
11. A. Prasad Sistla, Miloš Žefran, and Yao Feng. Monitorability of stochastic dynamical systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, pages 720–736, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
12. Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, and Erez Zadok. Runtime verification with state estimation. In Sarfraz Khurshid and Koushik Sen, editors, *Runtime Verification*, pages 193–207, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
13. D. Thorsley and D. Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Transactions on Automatic Control*, 50(4):476–492, April 2005.

A Diagnosability

A diagnoser is a policy that decides surely. We study this existence to know if a monitor exits for the system. We use this result when we do not consider the DFA in the hypothesis. In this case, test, if a diagnoser exists is PSPACE.

Definition 12. *A policy ρ is a* diagnoser *if and only if ρ decides almost surely.*

Lemma 5. (cf. [6, proposition 7]) *There exists a diagnoser if and only if ϵ is not confused.*

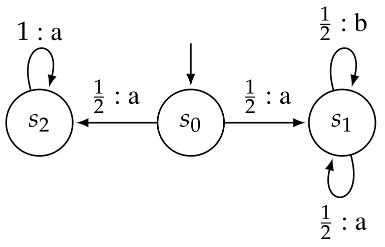

Fig. 10: An MC without diagnoser.

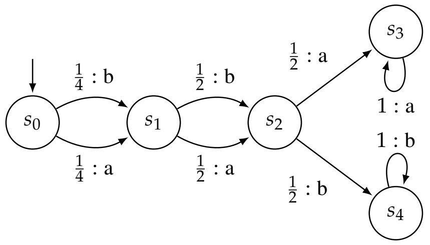

Fig. 11: A Deterministic MC.

Example 14. The MC in the Figure 10 does not have a diagnoser because ϵ is confused. For all $n \in \mathbb{N}$, we have $Pr(\{w \gtrsim a^n \mid w \in L\}) > 0$ and $Pr(\{w \gtrsim a^n \mid w \notin L\}) > 0$, so a^n is not deciding. If the MC takes the right transition first then almost surely it emits b at some point. Thus, $Pr(\{uw \mid u \sim aaa...\}) = \frac{1}{2}$. Hence, $Pr(\{\epsilon u \mid \epsilon u \text{ deciding}\}) < Pr(\{uw \mid u \sim \epsilon\})$ and ϵ is confused.

B Deterministic Markov Chain case

We extend our study on the Deterministic Markov Chain case. In this MC class, a state and a label identify each state. We hope obtains the better results than the general case. However, the policy $\rho_{ZeroDelay}$ is not optimal and we compute the policy in PSPACE.

Definition 13. *An MC $\mathcal{M} = (S, \Sigma, M, s_0)$ is called* Deterministic *the emitted letter and the current state identify the next state, i.e., there exists a function $\dashrightarrow : S \times \Sigma \to S$ such that $M(a)(s, s') > 0$ implies $s' = \overrightarrow{(s, a)}$.*

Example 15. The MC in the Figure 11 is a Deterministic MC. The state s_2 and the letter b identify the state s_4, but the state s_2 and the letter a identify the state s_3.

A non-Hidden MC is a Deterministic MC. In a non-Hidden MC, each letter identifies a state. Each letter and a state identify also a state.

Optimal policy As the general case, the policy $\rho_{ZeroDelay}$ is not optimal. For this class of MC, $\rho_{ZeroDelay}$ is also not optimal.

Theorem 7 (Policy $\rho_{ZeroDelay}$ is not optimal in the Deterministic MC case). *Given a DMC and a DFA. For all feasible policy without delay ρ, we do not have necessary: $Ex(C_{\rho_{ZeroDelay}}) \leq Ex(C_\rho)$.*

Idea of proof. We take the same counterexample (Figures 12 and 13) that for the general case (Theorem 3). It is a Deterministic MC. We prove the non-optimality of $\rho_{ZeroDelay}$ with the same proof of the Theorem 3.

Theorem 8. *Given an MC $\mathcal{M}$ and a DFA $\mathcal{A}$, $\rho_{ZeroDelay}$ can be computed in PSPACE.*

Proof. For all $a \in \Sigma$, we decide $L_{B_a} \subseteq L_{B_\perp}$ to compute the skip condition, by the Theorem 4. The inclusion of NFA languages is a problem PSPACE-complete [7]. Hence, evaluate the policy is a problem PSPACE.

C Language equivalent

Language equivalent in a certain DFA characterizes confusion. Consider the belief NFA $\mathcal{B}$. In the non-Hidden MC case, $\mathcal{B}$ becomes a DFA $\mathcal{B}'$ when we disallow $\perp$-transition. For $\mathcal{B}'$, we define the accepting states set by: $F = \{(s,q) | Pr_s(L_q) = 1\}$. We associate for each state (s,q) the language $L_{s,q} \subseteq \Sigma^*$. It is the language deciding by $\mathcal{B}'$ from the initial state (s,q).

Example 16. The Figure 6 is the DFA $\mathcal{B}'$ from the non-Hidden MC (Figure 1) and its DFA (Figure 2). In this Figure, states that are unreachable from (s_0,q_0) are not drawn here.

For this $\mathcal{B}'$, we define $L(s_1,q_1) = (ae)^*bb^*$.

Definition 14. *(s,q) and (s',q') are* language equivalent *when $(s,q) \approx (s',q')$ if and only if $L_{s,q} = L_{s',q'}$.*

Example 17. The Figure 6, (s_1,q_1) and (s_0,q_0) are not language equivalent. Indeed, $L_{s_1,q_1} = (ae)^*bb^*$ and $L_{s_0,q_0} = a(ae)^*bb^*$. Hence $L_{s_1,q_1} \neq L_{s_0,q_0}$.

Lemma 6. *(*cf. [6, Lemma 18]*) We compute the relation $\approx$ in polynomial time.*

Definition 15. *A belief B is* settled *when all $(s,q) \in B$ are language equivalent*

Example 18. In the Figure 6, the belief $B = \Delta((s_1,q_1),\perp)$ is not settled. Indeed, (s_3,q_f) and s_2,q_2 is in B. We have $L_{s_3,q_f} = b^*$ and $L_{s_2,q_2} = \emptyset$. Hence, $L_{s_3,q_f} \neq L_{s_2,q_2}$, and (s_3,q_f) and (s_2,q_2) are not language equivalent.

Lemma 7. *(*cf. [6, Lemma 19]*) A belief B is confused if and only if there is $a \in \Sigma$ such that $\Delta(B,a)$ is not settled.*

It follows that one can check in polynomial time whether a given belief is confused.

Example 19. In the Figure 6, the belief $B = \Delta((s_1,q_1),\perp)$ is not confused. Indeed, $\Delta(B,e) = (s_1,q_1)$, $\Delta(B,b) = (s_3,q_f)$ and $\Delta(B,c) = (s_2,q_2)$ are settled. By the lemma 7, B is not confused.

D Proofs

For the rest of the document, we fix these notations. For $n \in \mathbb{N}$, $v_\rho(n)(v)$ is the observation prefix obtained by the policy ρ after n step. Thus, $v_\rho(n)(v)$ is the length n prefixes of $\pi_\rho(v)$. We note $l_\rho(n)(w)$ the number of non-$\perp$ observations in $v_\rho(n)(w)$. We define $B_\rho(n)(w) = \Delta(B_0, v_\rho(n)(w))$. For all $a \in \Sigma_\perp$, $B_{\rho,n,a} = \Delta(B_\rho(n)(w), a)$. In the following, we suppress w in the notation to avoid clutter. If there is not doubt on the policy, we suppress ρ in the notation to avoid clutter.

D.1 Proof of lemma 3

Here is Lemma 3 from the main body:

Lemma 3. *For all belief B and $v \in \Sigma^*$, $\Delta(B, \epsilon)$ is deciding $\Rightarrow \Delta(B, v)$ is deciding.*

Proof. Let B a belief and $v \in \Sigma^*$ such that $\Delta(B, \epsilon)$ is positively deciding.

$\Delta(B, \epsilon)$ is positively deciding
$\Leftrightarrow$ B is positively deciding (by $\Delta(B, \epsilon) = B$)
$\Leftrightarrow \forall (s, q) \in B, Pr_s(L_q) = 1$ (definition of deciding)
$\Rightarrow Pr_s(\{w \gtrsim v \mid w \in L_q\}) = 1$ (definition of $Pr_s(L_q)$)
$\Leftrightarrow$ v is positively deciding from B (definition of deciding)
$\Leftrightarrow$ $\Delta(B, v)$ is deciding (lemma 2)

We have the same reasoning if $\Delta(B, \epsilon)$ is negatively deciding. We replace 1 by 0.
Hence, $\Delta(B, \epsilon)$ is deciding $\Rightarrow \Delta(B, v)$ is deciding.

D.2 Proof of lemma 4

Here is Lemma 4 from the main body:

Lemma 4. *For all belief B.*

1. *$L_B \subseteq \Sigma^*$ and $L_B \supsetneq \Sigma^*$.*
2. *$v \in L_B \Leftrightarrow \Delta(B, v)$ is deciding and $\Delta(B, v) \neq \emptyset$.*
3. *For all $a \in \Sigma$, $L^e_{B_a} \subseteq L^e_{B_\perp}$.*
4. *$L_{B_\perp} \subseteq \bigcup_{a \in \Sigma} L_{B_a}$.*

Proof. We prove all itemize. Let B a belief.

1. We have $L_B \subseteq \Sigma^*$ by union of two languages in Σ^*. Moreover, $L_B \supsetneq \Sigma^*$ because in the example 7, we have $L_B = e(ae)^*\{b, c\}\Sigma^*$. For $B = \{(s_0, q_0)\}$ and for $a^* \in \Sigma^*$, $a \notin L_B$. Hence, $L_B \subseteq \Sigma^\omega$ and $L_B \supsetneq \Sigma^\omega$.
2. Show $v \in L_B \Leftrightarrow (\Delta(B, v)$ is deciding $\wedge \Delta(B, v) \neq \emptyset)$.
   $v \in L_B \Leftrightarrow v \in L^e_B$ and $v \in L^d_B$ (definition of L_B)
   $\Leftrightarrow \Delta(B, v) \neq \emptyset$ and $\Delta(B, v)$ deciding (definition of L^e_B and L^d_B)
   Hence, $v \in L_B \Leftrightarrow (\Delta(B, v)$ is deciding $\wedge \Delta(B, v) \neq \emptyset)$.

3. Show that for all $a \in \Sigma$, $L^e_{B_a} \subseteq L^e_{B_\perp}$. Let $a \in \Sigma$ and $v \in L^e_{B_a}$.
$$\begin{aligned} v \in L^e_{B_a} &\Rightarrow \Delta(B_a, v) \neq \emptyset && \text{(definition of } L^e_B\text{)} \\ &\Rightarrow \Delta(B_\perp, v) \neq \emptyset && (\Delta(B_a,v) \subseteq \Delta(B_\perp,v) \text{ and } \Delta(B_a,v) \neq \emptyset) \\ &\Rightarrow v \in L^e_{B_\perp} && \text{(definition of } L^e_B\text{)} \end{aligned}$$
Hence, $L^e_{B_a} \subseteq L^e_{B_\perp}$
4. Show that $L_{B_\perp} \subseteq \bigcup_{a\in\Sigma} L_{B_a}$. Let $v \in L_{B_\perp}$.
$$\begin{aligned} v \in L_{B_\perp} &\Rightarrow v \in L^d_{B_\perp} && \text{(definition of } L_B\text{)} \\ &\Rightarrow \Delta(B_\perp, v) \text{ is deciding} && \text{(definition of } L^d_B\text{)} \\ &\Rightarrow \text{for all } a \in \Sigma, \Delta(B_a, v) \text{ is deciding} && \text{(definition of } B_\perp\text{)} \\ &\Rightarrow \begin{array}{l}\text{for all } a \in \Sigma, \Delta(B_a, v) = \emptyset \\ \text{or } \Delta(B_a, v) \neq \emptyset \text{ and deciding}\end{array} && (1)\ \text{(excluded middle)} \end{aligned}$$
Moreover, we have
$$\begin{aligned} v \in L_{B_\perp} &\Rightarrow v \in L^e_{B_\perp} && \text{(definition of } L_B\text{)} \\ &\Rightarrow \Delta(B_\perp, v) \neq \emptyset && \text{(definition of } L^e_B\text{)} \\ &\Rightarrow \text{exists } a \in \Sigma, \Delta(B_a, v) \neq \emptyset && \text{(definition of } B_\perp\text{)} \\ &\Rightarrow \text{exists } a \in \Sigma, \Delta(B_a, v) \neq \emptyset \text{ and deciding} && (1) \\ &\Rightarrow \text{exists } a \in \Sigma, v \in l_B && \text{(definition } L_B\text{)} \\ &\Rightarrow v \in \textstyle\bigcup_{a\in\Sigma} L_{B_a} \end{aligned}$$
Hence, $L^d_{B_\perp} \subseteq \bigcup_{a\in\Sigma} L_{B_a}$.

Remark 1. For all $a \in \Sigma$, we do not have necessarily $L_{B_a} \subseteq L_{B_\perp}$ or $L_{B_\perp} \subseteq L_{B_a}$. In the belief in Figure 6, let $B = \{(s_0, q_0)\}$. We have $L_{B_e} = (ae)^*b^* \cup (ae)^*c^*$ and $L_{B_\perp} = (ae)^*b^* \cup (ae)^*c^*$. As, $L_{B_e} = L_{B_\perp}$, $L_{B_a} \subseteq L_{B_\perp}$ and $L_{B_\perp} \subseteq L_{B_a}$. In the belief in Figure 5, let $B = \{(s_0, q_0)\}$. We have $L_{B_a} = ba^* \cup ab^*$ and $L_{B_\perp} = \{b, a\}aa^* \cup \{b, a\}bb^*$. So, $L_{B_a} \subsetneq L_{B_\perp}$ ($b \in L_{B_a}$ and $b \notin L_{B_\perp}$) and $L_{B_\perp} \subsetneq L_{B_a}$ ($a^* \notin L_{B_a}$ and $a^* \in L_{B_\perp}$).

D.3 Proof of theorem 2

Here is Theorem 2 from the main body:

Theorem 2 (**$\rho_{ZeroDelay}$ is feasible and without delays**)**.** *For all deciding prefix v, $\pi_{\rho_{ZeroDelay}}(v)$ is also deciding.*

Proof. Let v a deciding prefix. We define this claim to prove the theorem.

Claim. Let v a deciding prefix. For all v_1 such that $v = v_1v_2$ then $v_{\rho_{ZeroDelay}}(|v_1|)(v)v_2$ is deciding.

We prove this claim by induction on the length of v_1. We define $\mathcal{P}_n$: "If $v = v_1v_2$ with $|v_1| = n$, then $v_{\rho_{ZeroDelay}}(n)(v)v_2$ is deciding ".

Case 0 Let v where $v = v_1v_2$ with $|v_1| = 0$. We have $v_1 = \epsilon$ and $v = v_2$. Hence, $[v_{\rho_{ZeroDelay}}(0)(v)]v_2 = \epsilon v_2 = v$ is deciding by hypothesis. Hence, $\mathcal{P}_0$ holds.

Case $n + 1$ Let v where $v = v_1v_2$ with $|v_1| = n + 1$. We can also write v as $v = v'_1a_nv_2$ with $|v'_1| = n$ and $a_n \in \Sigma$.

- Suppose that $\rho_{ZeroDelay}$ observes a_n. Then $v_{\rho_{ZeroDelay}}(n+1)(v) = [v_{\rho_{ZeroDelay}}(n)(v)]a_n$. We have $[v_{\rho_{ZeroDelay}}(n+1)(v)]v_2 = [v_{\rho_{ZeroDelay}}(n)(v)]a_n v_2$. By $\mathcal{P}_n$, $[v_{\rho_{ZeroDelay}}(n)(v)]a_n v_2$ is deciding and$[v_{\rho_{ZeroDelay}}(n+1)(v)]v_2$ is deciding. Hence, $\mathcal{P}_{n+1}$ holds.
- Otherwise, suppose that $\rho_{ZeroDelay}$ skips a_n. Then $v_{\rho_{ZeroDelay}}(n+1)(v) = [v_{\rho_{ZeroDelay}}(n)(v)]\bot$. By $\mathcal{P}_n$, $[v_{\rho_{ZeroDelay}}(n)(v)]a_n v_2$ is deciding. As $\Delta(B_n, a_n v_2) = \Delta(B_{n,a_n}, v_2)$, $\Delta(B_{n,\bot}, v_2)$ is also deciding.
  By definition of $\rho_{ZeroDelay}$, it skips a_n because ($\forall a \in \Sigma, \forall w : \Delta(B_{n,a}, w) = \emptyset \vee \Delta(B_{n,a}, w)$ is not deciding $\vee\ \Delta(B_{n,\bot}, w)$ is deciding). In particular, if $\Delta(B_{n,a_n}, v_2)$ is deciding, then $\Delta(B_{n,\bot}, v_2)$ is also deciding.
  By definition of a deciding belief, we conclude that $[v_{\rho_{ZeroDelay}}(n+1)(v)]v_2$ is deciding. Hence, $\mathcal{P}_{n+1}$ holds.

That prove the claim: for all v_1 such that $v = v_1 v_2$ then $[v_{\rho_{ZeroDelay}}(|v_1|)(v)]v_2$ is deciding.

Let $v_1 = v$, $\pi_{\rho_{ZeroDelay}}(v) = [v_{\rho_{ZeroDelay}}(|v_1|)(v)]v_2 = [v_{\rho_{ZeroDelay}}(|v_1|)(v)]\epsilon = v_{\rho_{ZeroDelay}}(|v_1|)(v)$ (because $v = v_1$ and $v_2 = \epsilon$). By the claim $v_{\rho_{ZeroDelay}}(|v_1|)(v)v_2 = \pi_{\rho_{ZeroDelay}}(v)$ is deciding.

Hence, $\pi_{\rho_{ZeroDelay}}(v)$ is deciding.

D.4 Proof of theorem 3

Here is Theorem 3 from the main body:

Theorem 3 (**$\rho_{ZeroDelay}$ is not optimal).** *For all feasible policy without delays ρ, we do not have necessarily: $Ex(C_{\rho_{ZeroDelay}}) \leq Ex(C_\rho)$.*

Proof. In the general case, the policy $\rho_{ZeroDelay}$ is not optimal. Let this counterexample with the MC in the Figure 12 and its DFA (Figure 13).

Let ρ the policy without delay such that ρ observes the first letter. If it observes 0, it skips the two next letters. Its observes two letters to make a decision. Else, it observes the next three letters. Its observes four letters to make a decision.

Let $w = 0^\omega$ and apply the policies $\rho_{ZeroDelay}$ and ρ on the belief obtained from the MC and its DFA. We have $\pi_{\rho_{ZeroDelay}}(w) = \bot 000$ and $\pi_\rho(w) = 0\bot\bot 0$. We compute $Ex(C_{\rho_{ZeroDelay}})$ and $Ex(C_\rho)$. $\rho_{ZeroDelay}$ always observes three letters to decide a word: $Ex(C_{\rho_{ZeroDelay}}) = 3$. The policy ρ observes two letters when 0 come first and four letters otherwise. So, $Ex(C_\rho) = 4\frac{1}{4} + 2\frac{3}{4} = 2.5$. Hence, $Ex(C_{\rho_{ZeroDelay}}) \geq Ex(C_\rho)$. In function of the MC probabilistic distribution, the optimal can be not reach.

D.5 Proof of theorem 4

Here is Theorem 4 from the main body:

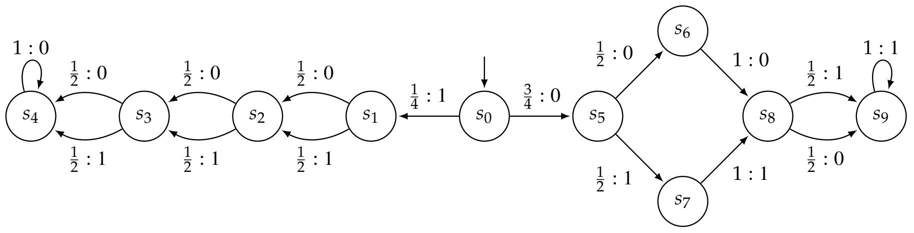

Fig. 12: MC for which $\rho_{ZeroDelay}$ is not optimal.

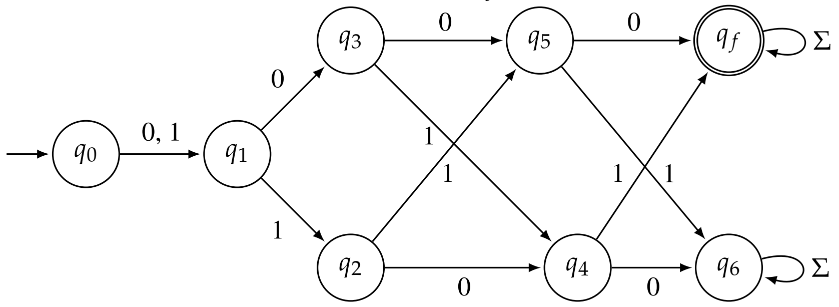

Fig. 13: DFA to control this MC.

Theorem 4. *For all belief B and $a \in \Sigma$, we have:*

$$L_{B_a} \subseteq L_{B_\perp} \Leftrightarrow (\forall v,\ \Delta(B_a, v) = \emptyset \vee (\Delta(B_a, v) \textit{ is deciding} \Rightarrow \Delta(B_\perp, v) \textit{ is deciding}))$$

Proof. Let B a belief and $a \in \Sigma$.

Suppose that $L_{B_a} \subseteq L_{B_\perp}$ and let $v \in \Sigma*$.

- Suppose that $v \notin L_{B_a}$.

  $v \notin L_{B_a} \Rightarrow v \notin L^d_{B_a}$ or $v \notin L^e_{B_a}$ (definition of L_B)

  $\Rightarrow$ $\Delta(B_a, v) = \emptyset$ or $\Delta(B_a, v)$ is not deciding (definition of L^e_B and L^d_B)

  $\Rightarrow$ $\Delta(B_a, v) = \emptyset \vee$ $(\Delta(B_a, v)$ is deciding $\Rightarrow \Delta(B_\perp, v)$ is deciding$)$

  Hence, v holds $(\Delta(B_a, v) = \emptyset \vee (\Delta(B_a, v)$ is deciding $\Rightarrow \Delta(B_\perp, v)$ is deciding$))$.
- Otherwise, suppose that $v \in L_{B_a}$.

  $v \in L_{B_a} \Rightarrow v \in L_{B_\perp}$ (hypothesis)

  $\Rightarrow \Delta(B_a, v)$ and $\Delta(B_\perp, v)$ are deciding (Lemma 4)

  $\Rightarrow \Delta(B_a, w)$ is deciding $\Rightarrow \Delta(B_\perp, w)$ is deciding

  Hence, v holds $(\Delta(B_a, v) = \emptyset \vee (\Delta(B_a, v)$ is deciding $\Rightarrow \Delta(B_\perp, v)$ is deciding$))$.

Reciprocally, suppose that $(\forall w,\ \Delta(B_a, v) = \emptyset \vee (\Delta(B_a, v)$ is deciding $\Rightarrow$ $\Delta(B_\perp, v)$ is deciding$))$ and let w holds the property such that $w \in L_{B_a}$.

- Suppose that $\Delta(B_a, v) = \emptyset$. By Lemma 4, $v \notin L_{B_a}$. We have a contradiction, so $\Delta(B_a, v) \neq \emptyset$.

- Otherwise, suppose that $\Delta(B_a, v) \neq \emptyset$.
  - Suppose that $\Delta(B_a, w)$ is not deciding. By Lemma 4, $w \notin L_{B_a}$. We have a contradiction because by hypothesis $w \in L_{B_a}$.
  - Otherwise, suppose that $\Delta(B_a, w)$ is deciding.
    $$\begin{aligned}\Delta(B_a, w) \text{ is deciding} &\Rightarrow \Delta(B_\perp, w) \text{ is deciding (hypothesis)} \\ &\Rightarrow w \in L_{B_\perp} \qquad\qquad \text{(Lemma 4)}\end{aligned}$$
    Hence, $w \in L_{B_\perp}$.

  Hence, $L_{B_a} \subseteq L_{B_\perp}$.

D.6 Proof of theorem 5

Here is Theorem 5 from the main body:

Theorem 5 **($\rho_{ZeroDelay}$ is optimal in the non-Hidden MC case).** *Given a non-Hidden MC and a DFA. For all feasible policy without delays ρ, we have $Ex(C_{\rho_{ZeroDelay}}) \leq Ex(C_\rho)$.*

Proof. Let ρ a policy without delay. We define this claim to prove the theorem.

Claim. For all w and all $n \in \mathbb{N}$:

$$l_{\rho_{ZeroDelay}}(n) \leq l_\rho(n) \text{ and } (B_{\rho_{ZeroDelay}}(n) \preceq B_\rho(n) \text{ or } l_{\rho_{ZeroDelay}}(n) < l_\rho(n))$$

We prove the claim by induction on n. Let w and define $\mathcal{P}_n$: "$l_{\rho_{ZeroDelay}}(n) \leq l_\rho(n)$ and $(B_{\rho_{ZeroDelay}}(n) \preceq B_\rho(n)$ or $l_{\rho_{ZeroDelay}}(n) < l_\rho(n)$".

Case 0 We have $l_{\rho_{ZeroDelay}}(0) = 0 = l_\rho(0)$ and $B_{\rho_{ZeroDelay}}(0) = B_0 \preceq B_0 = B_\rho(0)$. Hence, $\mathcal{P}_0$ holds.

Case $n+1$ Suppose that the property $\mathcal{P}_n$ holds.

- Suppose that $\rho_{ZeroDelay}$ does not observe the $(n+1)$st letter. In other word it skips the $(n+1)$st letter. Then, $l_{\rho_{ZeroDelay}}(n+1) = l_{\rho_{ZeroDelay}}(n)$.
  - Suppose $l_{\rho_{ZeroDelay}}(n+1) < l_\rho(n+1)$. Hence $\mathcal{P}_{n+1}$ holds.
  - Otherwise, by $\mathcal{P}_n$, $l_\rho(n+1) \leq l_{\rho_{ZeroDelay}}(n+1) = l_{\rho_{ZeroDelay}}(n) \leq l_\rho(n) = l_\rho(n+1)$. So, $l_\rho(n+1) = l_\rho(n)$. ρ does not observe this letter too. By $\mathcal{P}_n$, $B_{\rho_{ZeroDelay}}(n) \preceq B_\rho(n)$ and $B_{\rho_{ZeroDelay}}(n+1) = \Delta(B_{\rho_{ZeroDelay}}(n), \perp) \preceq \Delta(B_\rho(n), \perp) = B_\rho(n+1)$. Hence $\mathcal{P}_{n+1}$ holds.
- Otherwise, suppose that $\rho_{ZeroDelay}$ observes the $(n+1)$st letter a_{n+1}. From the definition of $\rho_{ZeroDelay}$ and a non-Hidden MC, it follows that $B_{\rho_{ZeroDelay}}(n+1)$ is settled and $B_{\rho_{ZeroDelay}}(n+1) \preceq B_\rho(n+1)$.
  - Suppose $l_{\rho_{ZeroDelay}}(n) < l_\rho(n)$. Then, $l_{\rho_{ZeroDelay}}(n+1) \leq l_\rho(n) \leq l_\rho(n+1)$. Hence, $\mathcal{P}_{n+1}$ holds.
  - Otherwise, by $\mathcal{P}_n$, $B_{\rho_{ZeroDelay}}(n) \preceq B_\rho(n)$.
    By definition of $\rho_{ZeroDelay}$, there exists a and w such that $\Delta(B_{\rho_{ZeroDelay}}, a, w) \neq \emptyset$ and $\Delta(B_{\rho_{ZeroDelay}}, a, w)$ is deciding and $\Delta(B_{\rho_{ZeroDelay}}, \perp, w)$ is not deciding. As $B_{\rho_{ZeroDelay}}(n) \preceq B_\rho(n)$,

$\Delta(B_{\rho,a},w) \neq \emptyset$ and $\Delta(B_{\rho,a},w)$ is deciding and $\Delta(B_{\rho,\perp},w)$ is not deciding. By the lemma 2, $[v_\rho(n)]\perp u$ is not deciding but $[v_\rho(n)]au$ is deciding. As, ρ is a policy without delay, ρ observes the next letters. Thus, $l_{\rho_{ZeroDelay}}(n+1) = l_{\rho_{ZeroDelay}}(n)+1 \leq l_\rho(n)+1 = l_\rho(n+1)$. Hence, $\mathcal{P}_{n+1}$ holds.

D.7 Proof of theorem 6

Here is Theorem 6 from the main body:

Theorem 6. *For all belief B such that $B_\perp$ is not confused and for all $a \in \Sigma$,*

$$L_{B_a} \subseteq L_{B_\perp} \Leftrightarrow (\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$$

Proof. Let B a belief such that $B_\perp$ is not confused and $a \in \Sigma$.

Suppose that $L_{B_a} \subseteq L_{B_\perp}$ and show that $(\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$.

- Suppose that $\epsilon \notin L_{B_a}$. Hence, $(\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$ holds.
- Otherwise, suppose that $\epsilon \in L_{B_a}$.
  $\epsilon \in L_{B_a} \Rightarrow \epsilon \in L_{B_\perp}$ $\quad (L_{B_a} \subseteq L_{B_\perp})$
  Hence, $(\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$ holds.

Reciprocally, suppose that $(\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$ and show $L_{B_a} \subseteq L_{B_\perp}$. Let $v \in L_{B_a}$.

- Suppose that $\epsilon \notin L_{B_a}$.
  $\epsilon \notin L_{B_a} \Rightarrow B_a$ is not deciding or $B_a = \emptyset$ (Lemma 4)
  $\Rightarrow B_a$ and $B_\perp$ are not deciding or $B_a = \emptyset$ (Lemma 4)
  - Suppose that $B_a = \emptyset$. So $L_{B_a} = \emptyset$, because $L^e_{B_a} = \emptyset$ (there are no enable prefix). Hence, $L_{B_a} \subseteq L_{B_\perp}$.
  - Otherwise, B_a and $B_\perp$ are not deciding. We recall that $B_\perp$ is not confused, by hypothesis.
    $B_\perp$ is not confused
    $\Rightarrow \forall a \in \Sigma, \Delta(B_\perp, a)$ is settle ([6, Lemma 19])
    $\Rightarrow \Delta(B_\perp, v_0)$ is settle $(v = v_0 v')$
    $\Rightarrow \forall (s,q),(s',q') \in \Delta(B_\perp, v_0), (s,q) \approx (s',q')$ (definition of settle)
    $\Rightarrow \forall (s,q) \in \Delta(B_a, v_0), \forall (s',q') \in \Delta(B_\perp, v_0), (s,q) \approx (s',q')$ $(B_a \subset B_\perp)$
    So, $\Delta(B_a, v_0)$ and $\Delta(B_\perp, v_0)$ describe the same language. As $v \in L_{B_a}$, $\Delta(\Delta(B_a,v_0),v')$ is deciding and not empty (where $v = v_0v'$). Hence $\Delta(\Delta(B_\perp,v_0),v')$ is not empty and is deciding.

  Hence, $L_{B_a} \subseteq L_{B_\perp}$.
- Otherwise, suppose that $\epsilon \in L_{B_\perp}$.
  $\epsilon \in L_{B_\perp} \Rightarrow B_\perp$ is deciding and $B_\perp \neq \emptyset$ (lemma 4)
  $\Rightarrow \Delta(B_\perp, v)$ is deciding and $B_\perp \neq \emptyset$ (1) (lemma 3)
  Moreover, by hypothesis $v \in L_{B_a}$.
  $v \in L_{B_a} \Rightarrow \Delta(B_a, v)$ is deciding and $\Delta(B_a,v) \neq \emptyset$ (lemma 4)
  $\Rightarrow \Delta(B_a, v)$ is deciding and $\Delta(B_\perp,v) \neq \emptyset$ (lemma 4)
  $\Rightarrow \Delta(B_\perp, v)$ is deciding and $\Delta(B_\perp,v) \neq \emptyset$ ((1))
  $\Rightarrow v \in L_{B_\perp}$ (definition of L_B)
  Hence, $L_{B_a} \subseteq L_{B_\perp}$.

Hence, $(L_{B_a} \subseteq L_{B_\perp} \Leftrightarrow (\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp}))$.

D.8 Proof of corollary 2

Here is Corollary 2 from the main body:

Corollary 2. *For all belief B and $a \in \Sigma$,*

$$(B_a = \emptyset \vee B_a \textit{ is deciding } \Rightarrow B_\perp \textit{ is deciding}) \Leftrightarrow (\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp})$$

Proof. Let B belief and $a \in \Sigma$.

Suppose that ($B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding) and show that ($\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp}$).

- Suppose that $B_a = \emptyset$. So, ϵ is not enable. By definition of L_B, $\epsilon \notin L_{B_a}$. Hence, ($\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp}$).
- Otherwise, suppose that (B_a is deciding $\Rightarrow B_\perp$ is deciding). Suppose that $B_a \neq \emptyset$. Else, we have in the previous case. As $B_a \subset B_\perp$, $B_\perp \neq \emptyset$. By definition of L_B, $\epsilon \in L_{B_\perp}$. Hence, $\epsilon \in L_{B_\perp}$.

Hence, ($\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp}$).

Reciprocally, suppose that ($\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp}$) and show that ($B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding).

- Suppose that $\epsilon \notin L_{B_a}$. By definition of L_B, $B_a = \emptyset$ or B_a is not deciding. Hence, ($B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding) holds.
- Otherwise, suppose that $\epsilon \in L_{B_\perp}$. By definition of L_B, $B_\emptyset \neq \emptyset$ and $B_\perp$ is deciding.
  - Suppose that B_a is not deciding. Hence, (B_a is deciding $\Rightarrow B_\perp$ is deciding) holds.
  - Otherwise, suppose that B_a is deciding. Hence, ($B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding) holds.

Hence, (($\epsilon \notin L_{B_a} \vee \epsilon \in L_{B_\perp}$) $\Leftrightarrow$ ($B_a = \emptyset \vee B_a$ is deciding $\Rightarrow B_\perp$ is deciding)).