Dead-end free single state multi-estimators for DES -the 2-estimator case

Camille Coquand, Xavier Pucel, Stéphanie Roussel, Louise Travé-Massuyès

Dead-end free single state multi-estimators for DES – the 2-estimator case

Camille Coquand[1,2], **Xavier Pucel**[1], **Stéphanie Roussel**[1], **Louise Travé-Massuyès**[2]
[1]ONERA, Université de Toulouse, France
e-mail: firstname.lastname@onera.fr
[2]LAAS-CNRS, Université de Toulouse, France
e-mail: louise@laas.fr

Abstract

Knowledge of the system state is key in autonomous decision making frameworks. To meet the run-time requirements and memory limitations that apply in this context, we propose an incremental estimation strategy that limits the number of estimates at each time step while still guaranteeing that dead-ends are not encountered. As this is not always achievable with a single-state estimator, we increase the number of estimators by defining the new notion of *multi-estimation*. In this paper, we consider discrete event systems and analyse the case of *2-estimation*, *i.e.* taking two single-state estimators to form a *2-estimator*. We first present a necessary condition for a 2-estimator being dead-end free and then derive a necessary and sufficient condition, both illustrated by experiments.

1 Introduction

Estimating the current state of an autonomous system is an essential task because of its impact on the strategy adopted by the autonomous system to fulfil its mission. For instance, when a drone is diagnosed with a critical fault, the associated strategy is to switch to a crash mode. With a precise estimation of the system state, it is possible to introduce degraded operational modes in which the system progressively loses performance as faults occur. State estimation is performed by a dedicated component called estimator, or diagnoser.

In this work, we assume that the system is modelled as a discrete event system (DES), for which [Sampath *et al.*, 1995] describes the construction of a diagnoser as a finite state machine (FSM) and defines the property of diagnosability (cf. [Grastien and Zanella, 2019] for a recent survey). [Shu *et al.*, 2007] investigates the notion of detectability and the construction of an observer that can be used as a diagnoser when the system is detectable. Recently, [Dague *et al.*, 2019] relaxes diagnosability by proposing manifestability that defines the weakest requirements on faults and observations for having a chance to identify on line fault occurrences.

When the system is partially observable, the number of possible states of the system at a given time can be extremely large. Even with symbolic representation techniques [Torta and Torasso, 2007], the limited capacity of autonomous systems in terms of memory requires to reduce the number of states returned by the estimator. In this work, we follow the idea of [Bouziat *et al.*, 2018] and adopt the strategy to keep only one state estimate at each time step. This estimation strategy brings promising results in terms of computation time [Bouziat *et al.*, 2018] and it significantly facilitates the task of the decision module. However it can lead to the estimator encountering dead-ends because a wrong estimate may be inconsistent with future observations. Note that using probabilistic diagnosers, as in many works [Fabre and Jezequel, 2010], does not solve the problem, the only solution being to backtrack [Kurien and Nayak, 2000], which is not compatible with real time constraints. We prefer the approach of [Bouziat *et al.*, 2019] that provides the conditions for a system to be single-state trackable (SST) and a dead-end free estimator synthesis procedure for such systems.

Because many real autonomous systems are not SST, this work proposes to extend this property to a larger class of systems. To do so, we propose to keep the idea of single state estimate but we increase the number of estimators, in defining the new notion of *multi-estimation*. In this paper, we analyse the case of two single-state estimators to form an estimator qualified as 2-estimator, hence proposing a 2-estimation strategy. The 2-estimation strategy still adapts to memory limitations by keeping no more than two estimates at each time step. Decision making must then include an arbitration process but it remains quite manageable.

As in [Roussel *et al.*, 2020], we model the system as a non-deterministic Moore machine, and the estimators as deterministic ones. We retrace our work by first presenting a necessary condition for a 2-estimator being dead-end free and then derive a necessary and sufficient condition.

The paper is organised as follows. In Section 2 , we recall the formal context of the study of [Roussel *et al.*, 2020] on which this work is based. In Section 3, we define 2-estimation and characterize dead-end free 2-estimators. Then we propose conditions to ensure dead-end free 2-estimators: Section 4 presents a necessary condition, then Section 5 presents a necessary and sufficient condition. Before concluding in Section 7, we illustrate the conditions with a set of experimental results in Section 6.

2 Systems and estimators formal framework

In this section, we recall the formal framework used in [Roussel *et al.*, 2020] to represent partially observable discrete event systems and estimators that return and keep in memory only one diagnosis at each time step. We also recall the dead-end issues raised by this type of estimators.

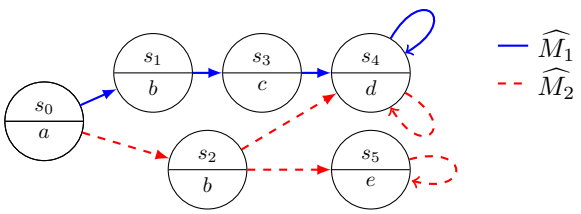

Figure 1: System M and two estimators $\widehat{M_1}$ and $\widehat{M_2}$.

2.1 System and language of a system

We model the discrete system dynamics as a Moore Machine with no input alphabet, and whose output alphabet consists in the observations.

Definition 1 (System)**.** *A system M is defined by a 5-tuple (S, s_0, O, Δ, obs) consisting of the following:*

- *a finite set of states S;*
- *an initial state $s_0 \in S$;*
- *a finite set of observations O (output alphabet of the Moore machine);*
- *a transition relation $\Delta \subseteq S^2$;*
- *an observation function $obs : S \to O$ mapping each state to its output.*

Without loss of generality, we consider that all the states of the system are reachable from the initial state s_0.

For a system M, we define $cands_M$ as the function from $S \times O$ to 2^S such that for all states s in S, for all o in O, $cands_M(s, o)$ represents the set of successors of s that have observation o in M. Formally, $\forall s \in S, \forall o \in O, cands_M(s, o) = \{t \in S | (s, t) \in \Delta \text{ and } obs(t) = o\}$.

A system M is *deterministic* if and only if for every $(s, o) \in S \times O$, we have $\#cands_M(s, o) \leq 1$.

Notation A state sequence seq is a list $s_0 \cdot s_1 \cdot \ldots \cdot s_{n-1} \in S^*$ where each s_i is a state in S; $|seq| = n$ is the length of the sequence and $seq[i] = s_i$ is the i^{th} state in the sequence; $last(seq)$ designates the last state of seq; if s is a state, $seq \cdot s$ is the sequence of length $|seq| + 1$ that begins with seq and ends with s. Similarly, we define an observation sequence $seq_{obs} \in O^*$, and extend the function obs to state sequences: if seq is a state sequence, $obs(seq)$ represents the observation sequence seq_{obs} such that for all $i < |seq|$, $seq_{obs}[i] = obs(seq[i])$.

Definition 2 (Language, observation language)**.** *The language associated with a system $M = (S, s_0, O, \Delta, obs)$ is the set of state sequences accepted by the system and starting with s_0. Formally $\mathcal{L}(M) = \{seq \in S^* | seq[0] = s_0 \text{ and for } i \in [1..|seq|[, (seq[i-1], seq[i]) \in \Delta\}$.*

The observation language is the language accepted by the system projected on the observations. Formally, $\mathcal{L}_{obs}(M) = \{obs(seq) | seq \in \mathcal{L}(M)\}$.

Example 1. *Let us consider the toy system M illustrated on Figure 1. Its set of states is $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$ with s_0 the initial state. All arrows represent transitions in Δ: for instance (s_0, s_1),(s_2, s_5), and (s_4, s_4) belong to Δ. The set of observations is $O = \{a, b, c, d, e\}$. The observation associated with each state is indicated in the lower part of the circle: for instance, $obs(s_0) = a$, $obs(s_3) = c$ and $obs(s_2) = b$.*
The language of M is the set of sequences accepted by M. For instance, s_0, $s_0 \cdot s_2 \cdot s_5$, and $s_0 \cdot s_1 \cdot s_3 \cdot s_4 \cdot s_4 \cdot s_4$ are sequences of $\mathcal{L}(M)$. Their corresponding observation sequences are a, $a \cdot b \cdot e$ and $a \cdot b \cdot c \cdot d \cdot d \cdot d$ and all belong to $\mathcal{L}_{obs}(M)$.
Because this system is partially observable, the observation b does not allow to distinguish s_1 and s_2.

2.2 Estimator and estimation function

Let us define what we call an estimation strategy and the resulting estimator for a system.

Definition 3 (Estimation strategy)**.** *An incremental single-state estimation strategy for a system $M = (S, s_0, O, \Delta, obs)$ is a function $estim : S \times O \to S$ such that for all s in S, for all o such that $cands_M(s, o) \neq \emptyset$, $estim(s, o)$ represents the estimated state of the system at time step n when the estimated state at time step $n-1$ is s and when o is observed at time step n.*

We impose the estimation strategy to be consistent both across time, *i.e.* $estim$ is a function, and with the system behaviour, *i.e.* $estim(s, o)$ belongs to $cands_M(s, o)$.

Definition 4 (Estimated sequence)**.** *Let M be a system, seq_{obs} be an observation sequence in $\mathcal{L}_{obs}(M)$ and $estim$ an estimation strategy for M. The estimated sequence for seq_{obs} is the state sequence $\widehat{seq} \in \mathcal{L}(M)$ such that $\widehat{seq}[0] = s_0$ and for all i in $[1..|seq_{obs}|[$, $\widehat{seq}[i] = estim(\widehat{seq}[i-1], seq_{obs}[i])$.*

An estimation strategy $estim$ for a system $M = (S, s_0, O, \Delta, obs)$ can be represented by a 5-tuple $\widehat{M} = (\widehat{S}, s_0, O, \widehat{\Delta}, obs)$, called an *estimator*, composed of the states and transitions of the system reachable with $estim$.

Definition 5 (Estimator)**.** *Let $M = (S, s_0, O, \Delta, obs)$ be a system and $estim$ an estimation strategy. The* estimator *induced by $estim$ is a system $\widehat{M} = (\widehat{S}, s_0, O, \widehat{\Delta}, obs)$ such that:*

- *$\widehat{S} \subseteq S$ is the set of states that belong to some estimated sequence for the given estimation strategy;*
- *$\forall (\widehat{s}, \widehat{t}) \in \widehat{S}^2$, $(\widehat{s}, \widehat{t}) \in \widehat{\Delta}$ if and only if $\widehat{t} = estim(\widehat{s}, obs(\widehat{t}))$.*

By construction, an estimator is a deterministic system.

Example 2. *Two estimators $\widehat{M_1}$ and $\widehat{M_2}$ are illustrated on Figure 1. They respectively correspond to estimation strategies $estim_1$ and $estim_2$ such that $estim_1(s_0, b) = s_1$ and $estim_2(s_0, b) = s_2$.*

2.3 Dead-end free estimator

As an estimator can be seen as a specific system, we can use Definition 2 to define its language and its observation language. Note that, by construction, if $\widehat{M}$ is an estimator of M, then $\mathcal{L}(\widehat{M}) \subseteq \mathcal{L}(M)$ and $\mathcal{L}_{obs}(\widehat{M}) \subseteq \mathcal{L}_{obs}(M)$.

Since an estimator only keeps in memory one state candidate for diagnosis, this candidate might not be the correct one, *i.e.* the estimated state is not the one in which the system really is. In such situations, if the system produces an observation that is not compatible with the estimated state, the estimator cannot explain the observation. We call such a scenario a *dead-end*.

Definition 6 (Dead-end)**.** *A* dead-end *for an estimator $\widehat{M}$ of a system M is an observation sequence that belongs to $\mathcal{L}_{obs}(M)$ but does not belong to $\mathcal{L}_{obs}(\widehat{M})$.*

An estimator $\widehat{M}$ without any dead-end is dead-end free*. In this case, we have $\mathcal{L}_{obs}(M) = \mathcal{L}_{obs}(\widehat{M})$.*

Example 3. *Let us consider the estimators illustrated on Figure 1. The observation sequences $a \cdot b \cdot d$ and $a \cdot b \cdot c$ are dead-ends for $\widehat{M}_1$ and $\widehat{M}_2$ respectively.*

We generalize the simulation as defined in [Roussel *et al.*, 2020] to a relation between any two systems that share the same set of observations. Informally, a system M_1 simulates another system M_2 if, at each time step, M_1 can exhibit the same set of future behaviours as M_2.

Definition 7 (Simulation for two systems)**.** *Let $M_1 = (S_1, init_1, O, \Delta_1, obs_1)$ and $M_2 = (S_2, init_2, O, \Delta_2, obs_2)$ be two systems. M_2 simulates M_1 denoted $M_1 \prec M_2$ if and only if there exists a relation $R \subseteq S_1 \times S_2$ such that:*

$$(init_1, init_2) \in R \tag{1}$$

$$\forall (s_1, s_2) \in R, obs_1(s_1) = obs_2(s_2) \tag{2}$$

$$\forall (s_1, s_2) \in R, \forall t_1 \in S_1 \text{ s.t. } (s_1, t_1) \in \Delta_1, \\ \exists t_2 \in S_2 \text{ s.t. } (s_2, t_2) \in \Delta_2 \text{ and } (t_1, t_2) \in R \tag{3}$$

In [Roussel *et al.*, 2020], the authors show that the equality of observation languages between a system and an estimator is equivalent to the existence of a simulation relation between them. The existence of such a relation can be computed polynomially ([Shukla *et al.*, 1996]).

Proposition 1 ([Roussel *et al.*, 2020])**.** *Let M be a system and $\widehat{M}$ an estimator for M. $\widehat{M}$ is dead-end free, i.e. $\mathcal{L}_{obs}(M) = \mathcal{L}_{obs}(\widehat{M})$, if and only if $\widehat{M}$ simulates M.*

Example 4. *From the example illustrated on Figure 1, there is no simulation relation between $\widehat{M}_1$ and M, because state s_2 of M cannot be simulated by any state in $\widehat{M}_1$. The same holds for $\widehat{M}_2$ because of state s_1. In fact, they can both encounter dead-ends.*

3 The 2-estimation problem

There are systems, such as the one illustrated on Figure 1, for which there does not exist any dead-end free estimator. However, for this system, running two single state estimators, $\widehat{M}_1$ and $\widehat{M}_2$ as shown on the figure, allows to avoid dead-ends. Hence, keeping a finite number of diagnostics in memory and not just one can solve the dead-end problem. We call such an estimation process *multi-estimation*. If the number of stored diagnosis stays reasonable, then memory limitations are still met. Decision making must then take into account the fact that the behaviour is only partially explained by each individual estimator. Yet we believe that an arbitration process such as prioritizing estimators is quite manageable.

In this section, we consider the case of two estimators to form an estimator qualified as *2-estimator*, hence analysing *2-estimation* strategies. Some properties allowing us to check whether 2-estimation solves the dead-end issue are proved. To do so, we first formally define 2-estimators and extend the concept of dead-end.

Definition 8 (2-estimator)**.** *A* 2-estimator *$\widehat{M}_1|\widehat{M}_2$ of a system M is formed from a set $\{\widehat{M}_1, \widehat{M}_2\}$ where $\widehat{M}_1$ and $\widehat{M}_2$ are estimators of M.*

We consider that the two estimators evolve independently. When an estimator produces a unique estimated state at each time step, a 2-estimator simply produces two states (that may be equal). This way, if one estimator encounters a dead-end, the other estimator may still continue to estimate. We define the language of a 2-estimator as the union of languages of the two estimators composing it.

Definition 9 (2-estimator language and observation language)**.** *Let M be a system and $\widehat{M}_1|\widehat{M}_2$ a 2-estimator for M. The language of $\widehat{M}_1|\widehat{M}_2$ is $\mathcal{L}(\widehat{M}_1|\widehat{M}_2) = \mathcal{L}(\widehat{M}_1) \cup \mathcal{L}(\widehat{M}_2)$.*

In the same way, we define the observation language of $\widehat{M}_1|\widehat{M}_2$ as $\mathcal{L}_{obs}(\widehat{M}_1|\widehat{M}_2) = \mathcal{L}_{obs}(\widehat{M}_1) \cup \mathcal{L}_{obs}(\widehat{M}_2)$.

Example 5. *The observation sequence $a \cdot b \cdot c \cdot d$ belongs to the observation language of $\widehat{M}_1|\widehat{M}_2$. It produces the estimated sequence $s_0 \cdot s_1 \cdot s_3 \cdot s_4$ which belongs to $\mathcal{L}(\widehat{M}_1|\widehat{M}_2)$.*

A 2-estimator cannot produce any diagnosis of the system if both estimators encounter dead-ends for a specific observation sequence that is consistent with the system. Such a situation is a dead-end for the 2-estimator, as formally defined below.

Definition 10 (Dead-end for a 2-estimator)**.** *Let M be a system and $\widehat{M}_1|\widehat{M}_2$ a 2-estimator of M. A sequence of observation sobs in $\mathcal{L}_{obs}(M)$ is a dead-end for $\widehat{M}_1|\widehat{M}_2$ if it is a dead-end for $\widehat{M}_1$ and for $\widehat{M}_2$.*

A 2-estimator $\widehat{M}_1|\widehat{M}_2$ is dead-end free *if $\mathcal{L}_{obs}(M) = \mathcal{L}_{obs}(\widehat{M}_1|\widehat{M}_2)$.*

4 A necessary condition for dead-end free 2-estimators

In order to check the existence of dead-ends for a 2-estimator $\widehat{M}_1|\widehat{M}_2$, we build a system that over-approximates the behaviour of the 2-estimator by "merging" $\widehat{M}_1$ and $\widehat{M}_2$. The idea behind this merged estimator is to check whether it accepts all observation sequences of the system. If not, such observation sequences are dead-ends for the 2-estimator.

The merged estimator is basically a system that contains every state and every transition of the two estimators of the 2-estimator.

Definition 11 (Merged estimator)**.** *Let $M = (S, s_0, O, \Delta, obs)$ be a system, $\widehat{M}_1 = (\widehat{S}_1, s_0, O, \widehat{\Delta}_1, obs)$ and $\widehat{M}_2 = (\widehat{S}_2, s_0, O, \widehat{\Delta}_2, obs)$ two estimators of M.*
We define the merged estimator as the system $merge(\widehat{M}_1, \widehat{M}_2) = (\widehat{S}_1 \cup \widehat{S}_2, s_0, O, \widehat{\Delta}_1 \cup \widehat{\Delta}_2, obs)$.

As the merged estimator contains every transition of both estimators $\widehat{M}_1$ and $\widehat{M}_2$, it accepts a superset of the union of the languages of the two estimators. However, it can accept observation sequences that belong neither to the language of one nor to the language of the other. In fact, we have $\mathcal{L}_{obs}(\widehat{M}_1|\widehat{M}_2) \subseteq \mathcal{L}_{obs}(merge(\widehat{M}_1, \widehat{M}_2)) \subseteq \mathcal{L}_{obs}(M)$. In terms of dead-ends, a dead-end for $merge(\widehat{M}_1, \widehat{M}_2)$ is also a dead-end for $\widehat{M}_1|\widehat{M}_2$. In other words, if $\widehat{M}_1|\widehat{M}_2$ is dead-end free, then the observation language of $merge(\widehat{M}_1, \widehat{M}_2)$ is equal to the language of the system.

Proposition 2. *Let M be a system and $\widehat{M_1}|\widehat{M_2}$ a 2-estimator for M. If $\widehat{M_1}|\widehat{M_2}$ is dead-end free, then $\mathcal{L}_{obs}(M) = \mathcal{L}_{obs}(merge(\widehat{M_1}, \widehat{M_2}))$.*

Proof. Let M be a system and $\widehat{M_1}|\widehat{M_2}$ a 2-estimator for M. If $\widehat{M_1}|\widehat{M_2}$ is dead-end free, then $\mathcal{L}_{obs}(M) = \mathcal{L}_{obs}(\widehat{M_1}|\widehat{M_2}) = \mathcal{L}_{obs}(\widehat{M_1}) \cup \mathcal{L}_{obs}(\widehat{M_2})$. By construction, the merged estimator accepts the union of the estimators languages. So we have $\mathcal{L}_{obs}(M) \subseteq \mathcal{L}_{obs}(merge(\widehat{M_1}, \widehat{M_2}))$. As $merge(\widehat{M_1}, \widehat{M_2})$ is a sub-system of M, we also have $\mathcal{L}_{obs}(merge(\widehat{M_1}, \widehat{M_2})) \subseteq \mathcal{L}_{obs}(M)$. □

The merged estimator of two deterministic systems can be non-deterministic. This is the case if $\widehat{M_1}$ and $\widehat{M_2}$ contain a different transition for the same state and the same observation. In order to have a condition using simulation similar to the one in Proposition 1, one requires a deterministic system. As the merged estimator of two estimators is not deterministic, we first determinise it following a classical approach where the determinised system states are sets of states of the original system.

Definition 12 (Determinisation of a system)**.** *Let $M = (S, s_0, O, \Delta, obs)$ be a system. The determinisation of M is the system $M^D = (S^D, \{s_0\}, O, \Delta^D, obs^D)$ where :*

- *S^D is a subset of 2^S where $\forall d \in S^D, \forall (s, s') \in d^2, obs(s) = obs(s')$*
- *$\forall (d, d') \in (S^D)^2$, $(d, d') \in \Delta^D$ if and only if $\exists s \in d$ and $s' \in d'$ such that $(s, s') \in \Delta$*
- *$obs^D : S^D \to O$ is defined as follows : $\forall d \in S^D, obs^D(d) = obs(s)$ where $s \in d$*

We show that a system and its determinisation have the same observation language.

Proposition 3. *Let M be a system. The determinised M^D has the same observation language. Formally, $\mathcal{L}_{obs}(M) = \mathcal{L}_{obs}(M^D)$.*

Proof. ($\subseteq$). Let $seq \in \mathcal{L}(M)$. We prove by recurrence that $\forall k \leq |seq|-1, \exists seq^D \in \mathcal{L}(M^D)$ such that $\forall i \leq k, seq[i] \in seq^D[i]$
($\supseteq$). Let $seq^D \in \mathcal{L}(M^D)$. We prove by recurrence that $\forall k \leq |obs^D(seq^D)| - 1, \forall s \in seq^D[k], \exists seq \in \mathcal{L}(M)$ s.t. $last(seq) = s$ and $\forall i \leq k$, $seq[i] \in seq^D[i]$. □

For a system M, we first prove that the determinised merged estimator of a 2-estimator for M has the same observation language as M if and only if there exists a simulation relation between the two systems.

Proposition 4. *Let M be a system and $\widehat{M_1}|\widehat{M_2}$ a 2-estimator. $merge(\widehat{M_1}, \widehat{M_2})^D$ simulates M if and only if $\mathcal{L}_{obs}(merge(\widehat{M_1}, \widehat{M_2})) = \mathcal{L}_{obs}(M)$.*

Proof. ($\Rightarrow$)Because $\widehat{M}^D$ simulates the system and by proposition 3, $\mathcal{L}_{obs}(merge(\widehat{M_1}, \widehat{M_2})) \subseteq \mathcal{L}_{obs}(M)$.
Then we prove by recurrence that if $\widehat{M}^D$ simulates M by the simulation relation R, then $\forall i \leq |seq| - 1, \exists Seq \in \mathcal{L}(\widehat{M}^D)$ such that $\forall l \leq i, (seq[l], Seq[l]) \in R$. Both seq and Seq produce the same observation sequence, because $\forall l \leq |seq| - 1, (seq[l], Seq[l]) \in R$, by equation 2, $obs(seq[l]) = obs^D(Seq[l])$, and then $\mathcal{L}_{obs}(M) \subseteq \mathcal{L}_{obs}(\widehat{M}^D)$.

($\Leftarrow$) Let us consider the following relation $R : \forall s \in S, d \in (\widehat{S}_1 \cup \widehat{S}_2)^D, (s, d) \in R$ iff $\exists seq_{obs} \in \mathcal{L}_{obs}(M)$, $seq \in \mathcal{L}(M), Seq \in \mathcal{L}(merge(\widehat{M_1}, \widehat{M_2})^D)$ *s.t.* $obs(seq) = obs^D(Seq) = seq_{obs}$, $s = last(seq)$, $d = last(Seq)$. We prove that R is a simulation relation in a similar way as proposition 1 is proven in [Roussel *et al.*, 2020]. □

Proposition 4 along with the inclusion of observation languages allows us to derive a necessary condition in the following corollary.

Corollary 1. *Let M be a system and $\widehat{M_1}|\widehat{M_2}$ a 2-estimator of M. If $\widehat{M_1}|\widehat{M_2}$ is dead-end free then $merge(\widehat{M_1}, \widehat{M_2})^D$ simulates M.*

5 A necessary and sufficient condition for dead-end free 2-estimators

In this section, we propose a necessary and sufficient condition for a 2-estimator to be dead-end free. To do so, we define a union operation $\oplus$ for two estimators which builds the *union estimator*. We show that the union estimator has the same language as the 2-estimator and propose a result similar to Proposition 1.

To represent the union estimator of a 2-estimator, we introduce the symbol $\perp$ that is used when one of the two estimators encounters a dead-end. Each state of the union estimator is either a pair of states from each estimator (*i.e.* a element of $S_1 \times S_2$) with the same observation or a pair composed of a state of one estimator and the element $\perp$. The initial state is the pair of the two initial states. A transition between two pairs of states is defined if the transition between the two first elements of the pair belongs to the first estimator and the same for the transition between the two second elements with respect to the second estimator, or if there is a transition only for one estimator, the second element of the successor's pair becomes $\perp$.

Definition 13 (Union estimator)**.** *Let $M = (S, s_0, O, \Delta, obs)$ be a system and $\widehat{M_1} = (\widehat{S}_1, s_0, O, \widehat{\Delta}_1, obs)$ and $\widehat{M_2} = (\widehat{S}_2, s_0, O, \widehat{\Delta}_2, obs)$ two estimators for M. The union of $\widehat{M_1}$ and $\widehat{M_2}$, denoted $\widehat{M_1} \oplus \widehat{M_2}$ is the system $(S_\oplus, (s_0, s_0), \Delta_\oplus, O, obs_\oplus)$ such that :*

- $S_\oplus = \{(s_1, s_2) | s_1 \in \widehat{S}_1, s_2 \in \widehat{S}_2, obs(s_1) = obs(s_2)\}$
  $\cup \{(s_1, \perp) | s_1 \in \widehat{S}_1\} \cup \{(\perp, s_2) | s_2 \in \widehat{S}_2\}$
- $\Delta_\oplus$ *is the smallest subset of* $S_\oplus^2$ *that contains*
  $\{((s_1, s_2), (s_1', s_2')) | (s_1, s_1') \in \widehat{\Delta}_1, (s_2, s_2') \in \widehat{\Delta}_2\}$
  $\cup \{((s_1, s_2), (s_1', \perp)) | (s_1, s_1') \in \widehat{\Delta}_1,$
  $cands_M(s_2, obs(s_1')) = \emptyset\}$
  $\cup \{((s_1, s_2), (\perp, s_2')) | (s_2, s_2') \in \widehat{\Delta}_2,$
  $cands_M(s_1, obs(s_2')) = \emptyset\}$
  $\cup \{((s_1, \perp), (s_1', \perp)) | (s_1, s_1') \in \widehat{\Delta}_1\}$
  $\cup \{((\perp, s_2), (\perp, s_2')) | (s_2, s_2') \in \widehat{\Delta}_2\}$
- $\forall s_1 \in \widehat{S}_1, \forall s_2 \in \widehat{S}_2 \cup \{\perp\}, obs_\oplus((s_1, s_2)) = obs(s_1),$
  $\forall s_2 \in \widehat{S}_2, obs_\oplus((\perp, s_2)) = obs(s_2)$

Example 6. *Figure 2 illustrates the union automaton $\widehat{M_1} \oplus \widehat{M_2}$ of the estimators defined in Figure 1. As (s_0, s_1)*

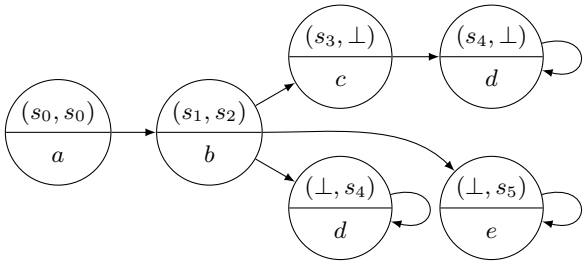

Figure 2: Union estimator $\widehat{M_1} \oplus \widehat{M_2}$ with estimators of Fig. 1

and (s_0, s_2) are respectively transitions of $\widehat{\Delta}_1$ and $\widehat{\Delta}_2$, $((s_0, s_0), (s_1, s_2))$ is a transition of $\Delta_\oplus$. There is no transition in $\widehat{\Delta}_2$ from state s_2 to a state with observation c so $((s_1, s_2), (s_3, \bot))$ is a transition of $\Delta_\oplus$.

We first show that the observation language of a union estimator is the same than that of the two-estimator's.

Proposition 5. *Let $M = (S, s_0, O, \Delta, obs)$ be a system and $\widehat{M_1} = (\widehat{S}_1, s_0, O, \widehat{\Delta}_1, obs)$ and $\widehat{M_2} = (\widehat{S}_2, s_0, O, \widehat{\Delta}_2, obs)$ two estimators for M. We have $\mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2}) = \mathcal{L}_{obs}(\widehat{M_1}) \cup \mathcal{L}_{obs}(\widehat{M_2})$.*

Proof. ($\subseteq$) Let us consider $seq_{obs} \in \mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2})$ with $n = |seq_{obs}|$. There exists $seq_\oplus \in \mathcal{L}(\widehat{M_1} \oplus \widehat{M_2})$ such that $obs_\oplus(seq_\oplus) = seq_{obs}$. For all $i < n$, let (t_i, u_i) denote $seq_\oplus[i]$.

- if $t_{n-1} \neq \bot$, consider $seq = t_0 \cdot \ldots \cdot t_{n-1}$. By construction of $\widehat{M_1} \oplus \widehat{M_2}$, $\forall i < n, t_i \neq \bot$ and $(t_i, t_{i+1}) \in \widehat{\Delta}_1$, which implies $seq \in \mathcal{L}(\widehat{M_1})$. As $obs(seq) = seq_{obs_\oplus}$, $seq_{obs} \in \mathcal{L}_{obs}(M_1)$;
- if $t_{n-1} = \bot$, then $u_{n-1} \neq \bot$, and we show that $seq = u_0 \cdot \ldots \cdot u_{n-1} \in \mathcal{L}(\widehat{M_2})$ and $seq_{obs} \in \mathcal{L}_{obs}(\widehat{M_2})$.

($\supseteq$) Let us consider $seq_{obs} \in \mathcal{L}_{obs}(\widehat{M_1}) \cup \mathcal{L}_{obs}(\widehat{M_2})$ with $n = |seq_{obs}|$. Because of symmetry between $\widehat{M_1}$ and $\widehat{M_2}$, we suppose that $seq_{obs} \in \mathcal{L}_{obs}(\widehat{M_1})$. There exists $seq \in \mathcal{L}(\widehat{M_1})$ such that $obs(seq) = seq_{obs}$.

- if seq_{obs} is not a dead-end for $\widehat{M_2}$, there exists $seq_2 \in \mathcal{L}(\widehat{M_2})$ such that $obs(seq_2) = seq_{obs}$. The sequence $(seq[0], seq_2[0]) \cdot \ldots \cdot (seq[n-1], seq_2[n-1])$ belongs to $\mathcal{L}(\widehat{M_1} \oplus \widehat{M_2})$, so $seq_{obs} \in \mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2})$.
- If seq_{obs} is a dead-end for $\widehat{M_2}$, we consider the greater integer $k < n$ such that $seq_{obs}[0] \cdot \ldots \cdot seq_{obs}[k-1])$ is not a dead-end for $\widehat{M_2}$ and $seq_2 = seq_2[0] \cdot \ldots \cdot seq_2[k-1])$ its associated sequence in $\mathcal{L}(\widehat{M_2})$. The sequence $(seq[0], seq_2[0]) \cdot \ldots \cdot (seq[k-1], seq_2[k-1]) \cdot (seq[k], \bot) \cdot \ldots \cdot (seq[n-1], \bot)$ belongs to $\mathcal{L}(\widehat{M_1} \oplus \widehat{M_2})$, so $seq_{obs} \in \mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2})$.

□

We now show that the union estimator is deterministic.

Proposition 6. *Let $M = (S, s_0, O, \Delta, obs)$ be a system and $\widehat{M_1} = (\widehat{S}_1, s_0, O, \widehat{\Delta}_1, obs)$ and $\widehat{M_2} = (\widehat{S}_2, s_0, O, \widehat{\Delta}_2, obs)$ two estimators for M. $\widehat{M_1} \oplus \widehat{M_2}$ is deterministic.*

Proof. Let us consider $o \in O$ and $(s_1, s_2) \in S_\oplus$. As $\widehat{M_1}$ is deterministic, there exists at most one $s_1' \in \widehat{S}_1$ such that $(s_1, s_1') \in \widehat{\Delta}_1$ and $obs(s_1) = o$. The same holds for $\widehat{M_2}$. If at least one transition exists in $\widehat{M_1}$ or $\widehat{M_2}$, only one transition corresponds in $\Delta_\oplus$. If no transition exist in $\widehat{M_1}$ or $\widehat{M_2}$, there is no corresponding transition in $\Delta_\oplus$. □

The next proposition proposes a necessary and sufficient condition for characterising dead-end free 2-estimators based on the simulation relation.

Proposition 7. *Let $M = (S, s_0, O, \Delta, obs)$ be a system and $\widehat{M_1} = (\widehat{S}_1, s_0, O, \widehat{\Delta}_1, obs)$ and $\widehat{M_2} = (\widehat{S}_2, s_0, O, \widehat{\Delta}_2, obs)$ two estimators for M. $\widehat{M_1} \oplus \widehat{M_2}$ simulates M if and only if $\mathcal{L}_{obs}(\widehat{M_1}|\widehat{M_2}) = \mathcal{L}_{obs}(M)$.*

Proof. ($\Rightarrow$) First, by Prop. 5 and Def. 9, $\mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2}) = \mathcal{L}_{obs}(\widehat{M_1}) \cup \mathcal{L}_{obs}(\widehat{M_2}) = \mathcal{L}_{obs}(\widehat{M_1}|\widehat{M_2})$. Because $\widehat{M_1}$ and $\widehat{M_2}$ are two estimators of M, we have $\mathcal{L}_{obs}(\widehat{M_1}) \cup \mathcal{L}_{obs}(\widehat{M_2}) \subseteq \mathcal{L}_{obs}(M)$, so $\mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2}) \subseteq \mathcal{L}_{obs}(M)$.
Let us suppose that $\widehat{M_1} \oplus \widehat{M_2}$ simulates M with the simulation relation R. We prove by recurrence that $\forall seq \in \mathcal{L}(M)$ with $n = |seq|$, $\forall k \leq n-1$, $\exists seq_\oplus \in \mathcal{L}(\widehat{M_1} \oplus \widehat{M_2})$ *s.t.* $\forall i \leq k, seq[i] \in seq_\oplus[i]$ and $(seq[i], seq_\oplus[i]) \in R$. From equation (2) of Def. 7, for every sequence of $\mathcal{L}(M)$, there exists a sequence of $\mathcal{L}(\widehat{M_1} \oplus \widehat{M_2})$ that has the same observation sequence, so $\mathcal{L}_{obs}(M) \subseteq \mathcal{L}_{obs}(\widehat{M_1} \oplus \widehat{M_2})$.
($\Leftarrow$) Let us consider the following relation R : $\forall s \in S, (s_1, s_2) \in S_\oplus, (s, (s_1, s_2)) \in R$ iff $\exists seq_{obs} \in \mathcal{L}_{obs}(M)$, $seq \in \mathcal{L}(M)$, $seq_\oplus \in \mathcal{L}(\widehat{M_1} \oplus \widehat{M_2})$ *s.t.* $obs(seq) = obs_\oplus(seq_\oplus) = seq_{obs}$, $s = last(seq)$, $(s_1, s_2) = last(seq_\oplus)$. We prove that R is a simulation relation in a similar way as Prop. 1 is proven in [Roussel *et al.*, 2020]. □

Example 7. *With M, $\widehat{M_1}$ and $\widehat{M_2}$ of Figure 1, the union estimator $\widehat{M_1} \oplus \widehat{M_2}$ in Figure 2 simulates M. In fact, the 2-estimator $\widehat{M_1}|\widehat{M_2}$ is dead-end free.*

6 Experiments

We have implemented the two conditions previously presented for checking the presence of dead-ends for 2-estimators. They are both based on an encoding of the simulation relation into SAT clauses, as described in [Shukla *et al.*, 1996]. We have used the Scala language from which Sat4j (SAT solver) can be called. ([Le Berre and Parrain, 2010]).

To generate random benchmarks, we use the SST system generation scheme presented in [Roussel *et al.*, 2020]. In order to get dead-end free 2-estimators, we first consider two SST systems M_1 and M_2 generated randomly. From $M_1 \oplus M_2$, we extract two subsystems N_1 and N_2 that have the same observation language as M_1 and M_2 respectively. Using the SMT encoding of [Roussel *et al.*, 2020], we synthesise two dead-end free estimators $\widehat{N_1}$ and $\widehat{N_2}$ for N_1 and N_2. The 2-estimator $\widehat{N_1}|\widehat{N_2}$ for the system $M_1 \oplus M_2$ should be dead-end free. In order to get 2-estimators with dead-ends, we generate systems as the union of three random SST systems and the estimators from only 2 of them. In total, we have 600 systems with 26 to 1973 states, for which half of the estimators have dead-ends.

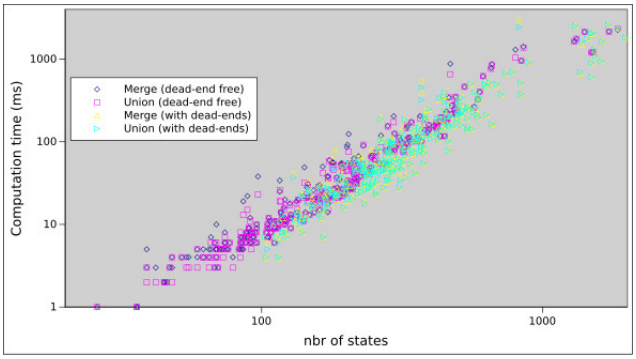

Figure 3: Computation time in function of the number of system states. Both scales are logarithmic.

We present the results of experiments on Figure 3. We distinguish dead-end free 2-estimators and estimators with dead-end on one side and the necessary condition (see Cor. 1) denoted *merge* and the necessary and sufficient one (see Prop. 7) denoted *union* on the other side. The computation time includes the time to perform operations on Moore machines and the time to check the simulation relation.

Figure 3 shows that the computation time is neither affected by the presence of dead-ends in estimators nor by the condition used for checking dead-ends. In fact, the operations performed on estimators (merge and determinisation on one side, and union on the other side) generate systems of the same magnitude. Most of the computation time is spent for checking the simulation relation, which explains similar efficiency. This behaviour might be linked to the way benchmarks are generated and more precisely to the fact that systems that are aggregated do not share many states. No benchmark has highlighted that the necessary condition *merge* is not sufficient, which also indicates that other benchmark generation schemes should be tested. Finally, the computation time does not blow up with the system size. This should be confirmed with a larger range of benchmarks.

7 Conclusion

This paper describes the problem of *multi-estimation*, focusing on 2-estimation. It provides two conditions, one necessary and the other necessary and sufficient, to check if a 2-estimator is dead-end free. The corresponding algorithms have been implemented and tested on randomly generated benchmarks, which shows the feasibility of the approach.

Future work includes a generalization to n-estimators. We suspect that as n grows, unions may have more states than determinised merged estimators, which would make the necessary condition more efficient than the necessary and sufficient condition.

Finally, another perspective is to follow the [Roussel *et al.*, 2020] approach and define n-Single State Trackability as the property of a system for which there exists a dead-end free n-estimator.

Acknowledgements

We would like to thank the cluster TrustMeIA[1] that funded the internship during which this work was performed.

[1]https://www.laas.fr/projects/trustmeia/

References

[Bouziat *et al.*, 2018] Valentin Bouziat, Xavier Pucel, Stéphanie Roussel, and Louise Travé-Massuyès. Preferential discrete model-based diagnosis for intermittent and permanent faults. In *Proceedings of the 29th International Workshop on Principles of Diagnosis*, 2018.

[Bouziat *et al.*, 2019] Valentin Bouziat, Xavier Pucel, Stéphanie Roussel, and Louise Travé-Massuyès. Single state trackability of discrete event systems. In *Proceedings of the 30th International Workshop on Principles of Diagnosis*, 2019.

[Dague *et al.*, 2019] Philippe Dague, Lulu He, and Lina Ye. How to be sure a faulty system does not always appear healthy?: Fault manifestability analysis for discrete event and timed systems. *Innovations in Systems and Software Engineering*, 2019.

[Fabre and Jezequel, 2010] Eric Fabre and Loïg Jezequel. On the construction of probabilistic diagnosers. In *WODES 2010 - 10th International Workshop on Discrete Event Systems*, volume 43, pages 229 – 234, Berlin, Germany, August 2010.

[Grastien and Zanella, 2019] Alban Grastien and Marina Zanella. Discrete-event systems fault diagnosis. In *Fault Diagnosis of Dynamic Systems*, pages 197–234. Springer, 2019.

[Kurien and Nayak, 2000] James Kurien and P. Pandurang Nayak. Back to the future for consistency-based trajectory tracking. In *Proceedings of the 17th AAAI Conference on Artificial Intelligence*, pages 370–377, 2000.

[Le Berre and Parrain, 2010] Daniel Le Berre and Anne Parrain. The SAT4J library, Release 2.2, System Description. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

[Roussel *et al.*, 2020] Stéphanie Roussel, Xavier Pucel, Valentin Bouziat, and Louise Travé-Massuyès. Model-based synthesis of incremental and correct estimators for discrete event systems. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*, 2020.

[Sampath *et al.*, 1995] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, 40(9):1555–1575, 1995.

[Shu *et al.*, 2007] Shaolong Shu, Feng Lin, and Hao Ying. Detectability of discrete event systems. *IEEE Transactions on Automatic Control*, 52(12):2356–2359, 2007.

[Shukla *et al.*, 1996] Sandeep Shukla, Daniel Rosenkrantz, Harry Iii, and Richard Stearns. The polynomial time decidability of simulation relations for finite state processes: A HORNSAT based approach. *Satisfiability Problem: Theory and applications*, 1996.

[Torta and Torasso, 2007] Gianluca Torta and Pietro Torasso. An on-line approach to the computation and presentation of preferred diagnoses for dynamic systems. *AI Communications*, 20(2):93–116, 2007.