On the cost of diagnosis with disambiguation

Loïc Hélouët, Hervé Marchand

On the cost of diagnosis with disambiguation

Loïc Hélouët, Hervé Marchand

INRIA Rennes, Campus de Beaulieu,
35042 Rennes Cedex, France
Email: loic.helouet@inria.fr, herve.marchand@inria.fr

Abstract. Diagnosis consists in deciding from a partial observation of a system whether a fault has occurred. A system is *diagnosable* if there exists a mechanism (a *diagnoser*) that accurately detects faults a finite number of steps after their occurrence. In a regular setting, a *diagnoser* builds an estimation of possible states of the system after an observation to decide if a fault has occurred. This paper addresses diagnosability (deciding whether a system is diagnosable) and its cost for safe Petri nets. We define an energy-like cost model for Petri nets: transitions can consume or restore energy of the system. We then give a partial order representation for state estimation, and extend the cost model and the capacities of diagnosers. Diagnosers are allowed to use additional energy to refine their estimations. Diagnosability is then seen as an energy game: checking whether disambiguation mechanisms are sufficient to allow diagnosability is in 2-EXPTIME, and one can also decide whether diagnosability under budget constraint holds in 2-EXPTIME.

1 Introduction

This paper addresses diagnosability of partially observable systems with disambiguation mechanisms, under energy constraints. We consider systems that can consume and produce energy, and which energy consumption can be modeled as weights attached to actions. In the standard diagnosis setting [17], faults are occurrences of particular faulty events (complex fault patterns have also been proposed in [14]). Systems under diagnosis are equipped with sensors (software probes or physical equipments) that can signal *some* state changes or occurrences of *some* actions, yielding partial observation of the system. The objective of diagnosis is to build monitors that receive observations from sensors and must raise alarms when a fault occurrence is certain. Cost of diagnosis has mainly been defined as the cost needed to exploit sensors [5, 18] in order to guarantee diagnosability. However, real life systems are not exclusively passive: when a fault is suspected, a monitor can perform tests (read the status of a variable, use a calculator) to leverage ambiguity on the status of a system. It is hence natural to consider *active* diagnosis, i.e. that monitors can perform additional costly actions to get information on the status of the system. The question in this active setting is then whether a non-diagnosable system is diagnosable with the help of additional tests, while satisfying energy constraints.

The model used in this paper is *finite safe Petri nets*, with observable and unobservable transitions, equipped with a cost model. Each action produces or consumes energy. The system starts with an initial energy provision. One can assume that the original model never exhausts its energy provision (this property can however be tested). Additional energy consuming tests can be used to reduce ambiguity on the current state of the system. In this setting, faults are a subset of transitions of the system, and they are considered permanent: once a faulty transition has been fired, the system remains faulty.

Observations are sequences of observable events. As we are working with *safe* Petri nets, diagnosis can obviously be recast in a labeled transition system setting. However, we use the Petri net structure to propose a compact way to represent the *state estimate* that a diagnoser can build after an observation. We use *observation guided unfolding* to find processes that may have produced an observation. Upon sensible restrictions, unfolding always terminates. One can also maintain finite state estimates along arbitrary long observations. We then define when a diagnosis can be produced by looking at properties of its processes. The exact current state of a system is not always precisely known, as several processes can correspond to the same observation. Additional ambiguity comes from the fact that diagnosers do not observe what has effectively occurred since the last observable event. Even with uncertainty on current state of a system, faults can be detected: a fault has occurred with certainty iff all processes of its observation guided unfolding contain a fault. No fault has occurred if all processes of the unfolding are fault free. If the unfolding contains both kind of processes, then it is said *ambiguous*. With this structure, a system is *diagnosable* iff it can not remain ambiguous for an arbitrary long time. Diagnosability of a net is a PSPACE-complete problem.

A natural question for non-diagnosable systems is whether increasing the power of diagnosers can make the system diagnosable. We define disambiguation functions, that bring additional information on the current state of the observed system, and can be used to reduce the set of possible states the system can be in. The disambiguation functions proposed in this work are simple: they provide information on the contents of a place (this models access to boolean variables), but disambiguation is not limited to this simple setting. We assume that tests cost energy, and hence have to be used parsimoniously. The cost of disambiguation can be easily integrated in the original cost model by assigning a negative weight to each test. Our *active diagnosis* setting is the following: immediately after an observation, diagnosers are allowed to use *one* disambiguation function to refine their state estimate. In addition, use of tests must not exhaust the system's energy. Diagnosability can hence be redefined as the possibility to make a system diagnosable with the help of tests without exhausting its energy. This question can be answered in terms of energy games with partial information. We build an arena that represents the behavior of the diagnosed system, the possible sets of state estimates that a diagnoser might build, and the remaining energy. Diagnosability with disambiguation and an upper bounded energy budget (ULWUB diagnosability) is equivalent to a partial observation co-Büchi game [7] between a system and a diagnoser: a system with energy constraints is not diagnosable iff it can impose arbitrary long ambiguity or energy exhaustion. As a consequence, ULWUB diagnosability is in 2-EXPTIME. Testing whether disambiguation can make a system diagnosable without energy consideration is already in 2-EXPTIME.

This paper is organized as follows: Section 2 introduces the main notations of the paper, and introduces a cost model. Section 3 defines a partial order setting for diagnosis. Section 4 introduces disambiguation techniques for our model, and shows that one can decide whether disambiguation makes a system diagnosable in 2-EXPTIME. Even when the cost of disambiguation is considered, the question of disambiguation without exhausting energy resources is in 2-EXPTIME. Section 5 compares our work with related approaches and discusses possible extensions. Due to lack of space, proofs are only sketched, and provided in Appendix.

2 Models and definition of the diagnosis problem

Definition 1. *A Petri net is a tuple $\mathcal{N} = (P, T, {}^\bullet(), ()^\bullet, \lambda, m_0)$, where P is a set of places, T is a set of transitions, ${}^\bullet() : T \to 2^P$ is a preset relation, and $()^\bullet : T \to 2^P$ is a postset relation. We consider labeled nets, i.e. equipped with a map $\lambda : T \to \Sigma$.*

A *marking* of net $\mathcal{N}$ is a function $M : P \to \mathbb{N}$. In the rest of the paper, we will only consider *finite* and *safe* Petri nets, i.e. such that for every reachable marking M and for every place $p \in P$, $M(p) \leq 1$. Hence, markings can be seen as subsets of marked places, and we will write $M \subseteq X$ when the set of marked places in M is contained in X. The *size* of $\mathcal{N}$ is simply $|P|$. We partition alphabet Σ into observable and unobservable actions, i.e. $\Sigma = \Sigma_o \uplus \Sigma_{uo}$. A transition $t \in T$ is *observable* iff $\lambda(t) \in \Sigma_o$, and *unobservable* otherwise. Intuitively, only a part of the events that occur during a run of the system is monitored and logged. This assumption is sensible: some parts of a system usually have to be considered as a black boxes, and remain unobserved. This partial observation setting can also be a design choice to maintain logs of reasonable sizes, or monitor systems with a limited number of sensors. We also consider a subset $\Sigma_f \subseteq \Sigma_{uo}$ of faulty actions, and say that $t \in T$ is *faulty* iff $\lambda(t) \in \Sigma_f$.

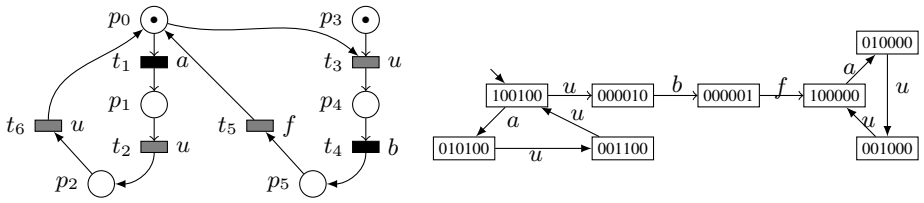

Fig. 1. A Petri net (left) and the associated LTS (right). Circles are places, marked places contain a token. Black rectangles are observable transitions and grey ones unobservable transitions. $\Sigma = \{a, b, f, u\}, \Sigma_o = \{a, b\}, \Sigma_f = \{f\}$.

Petri nets have an interleaved semantics, defined as follows: a transition $t \in T$ is *enabled* (denoted by $M[t\rangle$) from marking $M \subseteq P$ iff ${}^\bullet t \subseteq M$. Firing t from M results in a marking $M' = (M \backslash {}^\bullet t) \uplus t^\bullet$. This is denoted $M[t\rangle M'$. We will also write $M \xrightarrow{t} M'$ when $M[t\rangle M'$, and $M \xrightarrow{*} M'$ when there exists a sequence of transitions $M[t_1.t_2...t_n\rangle M'$. A *run* of $\mathcal{N}$ is a sequence $\rho = m_0 \xrightarrow{t_1} M_1 \dots M_n$. The set of *reachable configurations* from marking m_0 is the set $Reach(\mathcal{N}, m_0) = \{M \mid m_0 \xrightarrow{*} M\}$. The *labeling* of a run $\rho = m_0 \xrightarrow{t_1} M_1 \dots M_n$ is the word $w_\rho = \lambda(t_1).\lambda(t_2) \dots \lambda(t_n)$, and its *observation* is the word $\Pi_{\Sigma_o}(w_\rho)$, where $\Pi_{\Sigma_o}(.)$ is the usual *projection* erasing all letters of $\Sigma \setminus \Sigma_o$. We denote by $|\rho|_{\Sigma_o}$ the size of $\Pi_{\Sigma_o}(w_\rho)$. We will write $M \stackrel{a}{\Longrightarrow} M'$ iff there exists a run ρ from M to M' and $\Pi_{\Sigma_o}(w_\rho) = a$. Obviously, a safe Petri net defines a finite labeled transition system which states are $Reach(\mathcal{N})$. The LTS associated with Petri net $\mathcal{N}$ is $LTS_\mathcal{N} = (Q = Reach(\mathcal{N}), T, \delta, q_0 = m_0)$, where $\delta \subseteq Q \times T \times Q$ is a transition relation such that $(q, t, q') \in \delta$ iff $q \xrightarrow{t} q'$. $LTS_\mathcal{N}$ is of size in $O(2^{|P|})$.

Definition 2. *A* k-diagnoser *is a mechanism $\mathcal{D}$ that accepts observations and returns a value from $\{0, 1\}$, such that:*

– *for each non-faulty run* ρ, $\mathcal{D}\left(\Pi_{\Sigma_o}(\rho)\right) = 0$, *and*
– *for each faulty run such that at least* k *observable transitions have occurred since the first occurrence of a fault,* $\mathcal{D}\left(\Pi_{\Sigma_o}(\rho)\right) = 1$.

We say that a system is k-*diagnosable* if there exists no run ρ with k observations after a fault such that the observation of ρ has one faulty explanation and one non-faulty. A system is *diagnosable* if it is k-diagnosable for some $k \in \mathbb{N}$. A slightly different definition of diagnosability is proposed by [4]: it says that for every faulty run ρ, there exists a bound k_ρ such that a fault is detected at most k_ρ steps after it occurs. In a regular setting, both definitions coincide, but in a non-regular setting (e.g. for unbounded Petri nets) this is not always the case (as shown in [11]). It is frequently assumed that diagnosers are regular, i.e. one can compute an automaton that reads observations, and accepts all words such that $\mathcal{D}(w) = 1$. Upon acceptance, one can claim a fault occurred.

Definition 3. *A net is* diagnosable *if it is* k*-diagnosable for some* $k \in \mathbb{N}$.

In a regular setting, a necessary and sufficient condition [17] for diagnosability is existence of a bound K such that for every faulty execution ρ, for every execution $\rho' = \rho.\rho_1$ with $|\rho_1|_{\Sigma_o} \geq K$, every execution ρ'' such that $\Pi_{\Sigma_o}(\rho') = \Pi_{\Sigma_o}(\rho'')$ is faulty.

Proposition 1. *Deciding whether a Petri net* $\mathcal{N}$ *is diagnosable is* $PSPACE$*-complete. Furthermore, this can be decided in* $O(2^{2.|P|})$.

Proof (Sketch). One can search in PSPACE a pair of faulty/non-faulty runs with equivalent observation, that end on a loop (yielding infinite non-diagnosable executions). For the hardness part, one can encode regular languages intersection emptiness as the absence of obervationalley equivalent faulty/non-faulty runs. The exponential bound comes from the quadratic diagnosability decision procedure of [15] for automata, that can be applied to $LTS_{\mathcal{N}}$. □

In the rest of the paper, we will assume that the considered systems follow some energy consumption schemes: performing some actions consume energy, some others restore energy. The system starts with a known initial provision B_0. We also assume that systems have limited energy storage capacities (for instances batteries) and hence set an upper bound B_{max} for the amount of energy that can be stored by the system. A first question to consider is whether the system is properly designed and does not exhaust its energy provision, i.e. the remaining energy provision when the system does not use tests is always maintained above 0. For this, we define a cost model, and check that the system does not a priori consume more energy than it produces.

Definition 4. *A* cost model *for a Petri net* $\mathcal{N}$ *is a map* $\mathcal{C} : T \to \mathbb{N}$ *that associates integers to transitions of* T.

Definition 5. *Let* $\mathcal{N}$ *be a system with an initial energy budget* B_0 *and an upper energy bound* B_{max}*, and let* $\mathcal{C}$ *be a cost function. The* accumulated weight *with initial provision* B_0 *under weak upper bound* B_{max} *along a path* ρ *is denoted* $W_{B_0 \downarrow B_{max}}(\rho)$*, and is defined inductively as* $W_{B_0 \downarrow B_{max}}(\rho) = r_{|\rho|}$*, with* $r_0 = B_0$*,* $r_{i+1} = \min(r_i + \mathcal{C}(t_{i+1}), B_{max})$*. A net* $\mathcal{N}$ *satisfies the* universal lower-weak-upper-bound problem (UL-WUB) *iff, for every run* ρ *starting from* m_0 *we have* $0 \leq W_{B_0 \downarrow B_{max}}(\rho) \leq B_{max}$.

Informally, the accumulated weight along a run is the initial energy provision decreased by the cost of each action at every step in ρ, and bounded at each step by the maximal amount of energy the system can accumulate. If a system is well designed, and abstracting away external energy losses such as batteries aging, it should be able to run forever without exhausting its energy budget. This property can be easily verified. It was already shown that the ULWUB problem is polynomial for weighted automata ([3]), and consists in detecting lassos ending with cycles of negative weight. This result applies to $LTS_{\mathcal{N}}$, but we can be more precise. Assuming that the upper bound for the energy budget can be encoded by an integer smaller than $2^{|P|}$ we have:

Proposition 2. *The ULWUB problem is in PSPACE (w.r.t. the number of places in $\mathcal{N}$).*

Proof (Sketch). We can reuse the techniques proposed by [3], i.e. detect paths of length $< 2^{|P|}$ that end with a negative energy provision, or lassos that end with a cycle of negative cumulated weight. One does not need to explore paths of length greater than $2^{|P|}$, nor lassos of length $2^{|P|+1}$. This exploration can be performed nondeterministically by a search that memorizes at most 2 markings of the system and maintains energy budgets with $O(|P|)$ bits. This can hence be done in PSPACE.□

3 Diagnosis with processes of a net

The semantics of Petri nets can also be given in a non-interleaved setting with *processes*, a partially ordered representation of transitions firings.

Definition 6. *A* process *of a net $\mathcal{N} = (P, T, {}^\bullet(), ()^\bullet, \lambda_{\mathcal{N}}, m_0)$ is a tuple $\varepsilon = (E, B, \lambda)$, where E is a set of events, B is a set of conditions, and $\lambda : E \to \Sigma$ is a labeling of events. An* event *is a pair of the form $e = (X, t)$, where $X \subseteq B$ is a set of conditions needed to execute e, and t a particular transition. Given an event $e = (X, t)$, we denote by $t(e) = t$ the transition e refers to. To be consistent with $\mathcal{N}$, we have $\lambda(e) = \lambda_{\mathcal{N}}(t(e))$. A* condition *is a pair of the form $b = (e, p)$, where $e \in E$ and $p \in P$ is a reference to a place of $\mathcal{N}$. We denote by $Place(b)$ the place referred to in b.*

The intuitive understanding of processes is that events are occurrences of transitions firing, and conditions places which contents is consumed/produced when firing a transition. Processes define a partial ordering (antisymmetric, transitive, reflexive relation) among their elements. We write $x \prec y$ when $x = (e, p)$ is a condition and $y = (X, t)$ an event such that $x \in X$, or when $x = (X, t)$ is an event and y a condition of the form $y = (x, p)$. This way, a process also defines a partial order among its events: we write $e \leq_\varepsilon e'$ when $e \prec^* e'$. Given an element x in a process, the set of its predecessors is denoted $\downarrow x$ and is the set $\downarrow x = \{y \in E \cup B \mid y \prec^* x\}$. Similarly, the set of successors of x is denoted $\uparrow x$ and is the set $\uparrow x = \{y \in E \cup B \mid x \prec^* y\}$. An element x is *minimal* (resp. *maximal*) in a process iff $\downarrow x = \{x\}$ (resp. $\uparrow x = \{x\}$). Minimal and maximal elements in processes are conditions. We denote by $min(\varepsilon)$ the set of minimal places in ε and by $max(\varepsilon)$ the set of maximal places in ε. Processes are *conflict-free*: for every pair of distinct events $e, e' \in E$, we have ${}^\bullet e \cap {}^\bullet e' = \emptyset$. Processes are also *join-free*: $\forall e \neq e' \in E$, $e^\bullet \cap e'^\bullet = \emptyset$.

Processes have been well studied (see for instance [9]) but for completeness, we give below an inductive construction technique for processes of a net $\mathcal{N}$, starting from marking

m_0. We assume a dummy event $\perp$, and create a set of conditions $B_0 = \{(\perp, p) \mid p \in m_0\}$. The initial process is $\varepsilon_0 = (\emptyset, B_0, \lambda_0)$, where λ_0 is the empty map. Then, we iterate the following construction for each process $\varepsilon_i = (E_i, B_i, \lambda_i)$: we compute the set Max_i of maximal conditions in ε_i. Max_i corresponds to the state (marking) of the system once all transitions appearing in process ε_i have been executed. An occurrence of transition t can be appended to ε_i as soon as ${}^\bullet t \subseteq Place(Max_i)$. Intuitively, after executing ε_i, the places needed by t to fire are filled. Note that more than one transition can satisfy this condition. When t can be appended to ε_i we can define $e_{i+1} = (Max_i \cap Place^{-1}({}^\bullet t), t)$, the event consuming conditions from Max_i that are instances of places in ${}^\bullet t$, and build the process $\varepsilon_{i+1} = (E_i \cup e_{i+1}, B_i \cup \{(e_{i+1}, p) \mid p \in t^\bullet\}, \lambda_i \cup (e_{i+1} \to \lambda(t)))$.

A *linear extension* of a process $\varepsilon = (E, B, \lambda)$ is a sequence of events $e_1 \dots e_n$ such that $n = |E|$ and for every $i < j$, $e_j \not<^* e_i$. A *linearization* of ε is a word $w = a_1.a_2 \dots a_n$ such that there exists a linear extension u of ε with $\lambda(u) = w$. Intuitively, linearizations of ε are words that could be logged sequentially during execution of ε. Processes are a compact way to represent executions of Petri nets. For every run $\rho = m_0 \xrightarrow{t_1} m_1 \dots \xrightarrow{t_n} m_n$ of $\mathcal{N}$, there exists a unique process ε_ρ obtained by appending successively $t_1, \dots t_n$. Considering conditions as places, and events as transitions, processes are occurrence nets, i.e. an acyclic, join-free and conflict free type of net. We can hence safely talk about runs of a process and denote by $Confs(\varepsilon) \subseteq 2^P$ the configurations that can be reached during executions of $\mathcal{N}$ represented by ε. Figure 2 shows two processes for the net of Figure 1, with $\Sigma_o = \{a, b\}, \Sigma_{uo} = \{u, f\}$ and $\Sigma_f = \{f\}$. Conditions are represented as circles, and events as rectangle. Keeping the same convention as for Petri nets, we use black rectangles when events are occurrences of observable transitions, and grey rectangles otherwise. The left process corresponds to an observation $b.a$ and contains a fault, the right one corresponds to an observation $a.b$, and contains no fault.

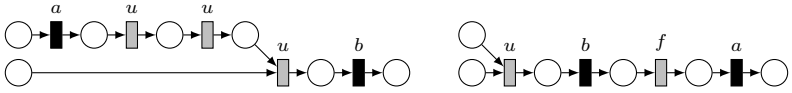

Fig. 2. Two processes for the net of Figure 1.

Definition 7. *Let $\Sigma = \Sigma_o \uplus \Sigma_{uo}$ be a finite alphabet, $\mathcal{N}$ be a Petri net labeled by Σ, and $w \in \Sigma_o^*$ be an observation. A process ε of $\mathcal{N}$ is an* explanation *of w iff $\varepsilon = \varepsilon_\rho$ for some run $\rho = m_0 \xrightarrow{t_1} m_1 \dots \xrightarrow{t_n} m_n$ of $\mathcal{N}$ such that $\Pi_{\Sigma_o}(w_\rho) = w$.*

Hence, a process ε is an explanation of an observation w iff w is the projection a linearization $a_1 \dots a_{|w|}$ of ε on Σ_o. Note that explanations can contain an arbitrary number of occurrences of unobservable transitions occurring after or concurrently with the last observable transition. Our objective is to define mechanisms that detect faults from partial observations after a finite number of steps of the system. Moreover, we want these mechanisms to run with finite memory. This is not always possible as shown in [4] for general Petri nets. However, for safe Petri nets, sensible restrictions allow diagnosers to memorize finite suffixes of processes.

3.1 Building an explanation for an observation

Let $w \in \Sigma_o^*$ be the observation of some run of $\mathcal{N}$. We can build inductively a set $\mathbb{E}_w$ of processes that "explain" w as follows. We starts from a set $\mathbb{E}_w^0$ that contains process $\varepsilon_0 = (\emptyset, B_0, \lambda_0)$. At each step, starting from a set of processes $\mathbb{E}_w^i$, one can select a process ε, and append to it either an unobservable transition, or a transition labeled by the next letter to explain in w. The construction stops when reaching a set of processes $\mathbb{E}_w^n$ that can not be extended without adding more observable events than in w.

As w is an observation of $\mathcal{N}$, $\mathbb{E}_w$ contains at least one process. Due to concurrency and choices, several transitions can usually be appended to a process. Hence the observation guided construction above is non-deterministic. Every unobservable transition can be appended to $\varepsilon \in \mathbb{E}_w^i$ if it is allowed from configuration $Max(\varepsilon)$, and an observable transition can be appended if it carries the label of the next unexplained action in w. There can be several occurrences of such transitions in a labeled net. As the set of appendable transitions may contain conflicting transitions $\mathbb{E}_w$ can contain more than one process. We give a complete algorithm for the construction of $\mathbb{E}_w$ in Appendix D.

Addition of an unobservable transitions to a process $\varepsilon \in \mathbb{E}_w^i$ is not conditioned by w. Hence, in general, process construction (w.r.t w) needs not stop. Indeed, one can append successively an arbitrary number of unobservable transitions, and without restriction, the set of explanations $\mathbb{E}_w$ for observation $w \in \Sigma_o^*$ may not be finite. We hence define the following restriction:

Definition 8. *A Petri net $\mathcal{N} = (P, T, {}^\bullet(), ()^\bullet, \lambda)$ with observable alphabet $\Sigma_o \subseteq \Sigma$ is* boundedly silent *iff there exists a bound $K \in \mathbb{N}$ such that for every marking $M \in Reach(\mathcal{N})$, there exists no process $\varepsilon = (E, B)$ that starts at M and such that $\lambda(t(E)) \subseteq (\Sigma \setminus \Sigma_o)$ and $|\varepsilon| > K$.*

Requiring boundedly silent nets is a sensible assumption asking that systems fire an observable event regularly. A similar notion exists for diagnosis of systems described with automata: it requires that systems have no unobservable cycle.

Proposition 3. *Let $\mathcal{N}$ be a boundedly silent Petri net, with bound K, and let $w \in \Sigma_o^*$ be an observation. Then, $\mathbb{E}_w$ is finite, and contains processes built in at most $(|w| + 1).K$ steps.*

Proof (sketch). We show by induction on the length of w (see Appendix A) that the inductive construction of processes terminates, and is sound and complete.

In Appendix D, we give an effective algorithm $Unfold(\mathcal{N}, w, m_0)$ to build $\mathbb{E}_w$ from a boundedly silent net $\mathcal{N}$ starting from marking m_0. The algorithm proceeds inductively and returns all explanations of w provided by $\mathcal{N}$. Every process ε_w^i in $\mathbb{E}_w$ has exactly $|w|$ observable transitions, and every observable transition of ε_w^i can be associated a letter of w that it explains. Note however that processes are "saturated" by appending all unobservable transitions that may have occurred without generating observations. Hence, some processes built by our algorithm contain unobservable transitions that are not *needed* to explain observation w. We can use the same algorithm to build a set of silent processes depicting maximal unobservable behaviors, starting from any marking by calling $Unfold(\mathcal{N}, \epsilon, M)$.

If all processes in $\mathbb{E}_w$ contain faulty processes, then one can claim without error that a fault has occurred. Similarly, if all processes in $\mathbb{E}_w$ are non-faulty, then one can claim that the observed behavior is non-faulty. If $\mathbb{E}_w$ is composed of faulty and non-faulty processes, then ambiguity remains on whether a fault occurred. This is a standard setting in diagnosis. Another frequent assumption in diagnosis is that systems are stopped immediately after occurrence of an observable action. Our setting slightly differs, as saturating processes means that one considers executions that have not yet produced an additional observation after w. Though this does not make a huge difference in decidability or algorithms, this setting seems more natural, especially in a context where unobserved additional transitions might be concurrent with the occurrence of the last observable ones. Let us comment this situation: let $\varepsilon \sqsubseteq \varepsilon'$ be two explanations of w. Supposing that ε is the actual behavior of $\mathcal{N}$ that led to observation w, then one can not decide whether some events in $\varepsilon' \setminus \varepsilon$ have occurred or not. This is a new source of ambiguity: if ε contains no faulty transition, and ε' is faulty, then ambiguity arises from the fact that one can not yet decide whether this faulty transition has already occurred or not. Note also that future occurrence of a fault after ε is not mandatory either, as after ε, $\mathcal{N}$ can still execute a non-faulty process ε'' such that $\varepsilon \sqsubseteq \varepsilon''$ instead of ε'.

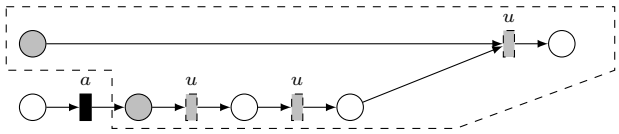

Fig. 3. A process for the net of of Fig. 1 wrt observation a. The frontier is denoted by greyed conditions. The summary is the set of events and conditions in the dashed zone.

3.2 Summaries

Building explanations online with observations results in ever growing processes. However, remembering a whole process is not needed for fault detection. In this section, we show that the important information to keep from a process is whether a fault has occurred, and the maximal conditions after observable events that must have happened in this process. It suffices to find the possible configurations of a system after an observation. This information can be memorized as a finite *summary*:

Definition 9. *Let $\varepsilon = (E, B, \lambda)$ be a process explaining $w \in \Sigma_o^*$. An event e in ε* must have occurred *if $\lambda(e) \in \Sigma_o$, or $\exists f, e \leq f$ and f must have occurred. We denote by $Must(\varepsilon)$ the set of events that must have occurred in ε. The* silent part *of ε is the restriction of ε to events in $E' = E \setminus \downarrow Must(\varepsilon)$ and conditions in ${}^\bullet E' \cup E'^\bullet$. The execution frontier $Frontier(\varepsilon)$ of process ε is the set of minimal places in the silent part of ε. The* summary *of ε is denoted $Sum(\varepsilon)$, and is the restriction of ε to $\uparrow Frontier(\varepsilon)$.*

Let us give the intuition behind the notions of frontier and summary of a process. A frontier is a possible configuration that the system reaches when executing the smallest

possible subset of events containing all observable events needed to explain w. This does not mean that after observing w the system is necessarily in this state, as unobservable transitions may have occurred concurrently with the last observable actions of w. Summaries capture parts of executions that *may* have occurred. A summary is also a process, starting from $Frontier(\varepsilon)$. Considering processes as standard nets, we can prove that $Frontier(\varepsilon)$ is a marking of ε and hence also that $Place(Frontier(\varepsilon))$ is a marking of $\mathcal{N}$ (see proof in Appendix E). Hence, up to a relabeling of minimal conditions in the silent summary $\varepsilon_S = Sum(\varepsilon)$, we have that ε_S is equivalent to some process in $Unfold(\mathcal{N}, \epsilon, Frontier(\varepsilon))$. Moreover, remembering silent parts of processes suffices to describe the set of possible configurations a system might be in after an observation.

Proposition 4. *Let ε be an explanation of observation $w \in \Sigma_o^*$, and ε_S be its summary. Then for every marking M of ε_S, $Place(M)$ is a marking that is reachable via a run ρ of $\mathcal{N}$ such that $\lambda(\rho) = w$.*

Proposition 5. *Let ε be an explanation of word $w \in \Sigma_o^*$, and M be its frontier. Then $\varepsilon' \sqsupseteq \varepsilon$ is an explanation of $w.a \in \Sigma_o^*$ iff there exists $\varepsilon_a \in Unfold(\mathcal{N}, a, M)$ such that $\varepsilon' = \varepsilon \cup \varepsilon_a$ (where union of processes means union component wise).*

When performing diagnosis, the important information to memorize in processes is whether a fault has occurred, or may have occurred, and the configurations the system might be in after some observation. Note that the set of summaries is an explanation of an observation is finite. Note also that $Frontier(\varepsilon)$ is the minimal set of places in $\varepsilon_S = Sum(\varepsilon)$. In the rest of the paper, we will write $\varepsilon_S \xrightarrow{a} \varepsilon'$ if ε_S is a summary with frontier M, and $\varepsilon' \in Unfold(\mathcal{N}, a, M)$ (one can execute observable action a from some configuration in the state estimation contained in ε_S and obtain a process ε'. Note that ε' is not a summary, but can be projected to obtain $\varepsilon'_S = Sum(\varepsilon')$. Note also that in general $\varepsilon' \not\sqsupseteq \varepsilon$, as the execution chosen from M may differ from the possible execution depicted by ε_S. The common part between ε_S and ε'_S is $\varepsilon_S \cap \varepsilon'$, and the part of ε' that does not appear in ε_S is denoted $\varepsilon' \setminus \varepsilon_S$.

A *possible state* after an observation $w \in \Sigma_o^*$ is a pair (V, ε_S) where ε_S is a summary of some explanation ε of w, and V is a tag from $\{F, N, A\}$ called a *verdict*. Tag F stands for faulty, N stands for non-faulty, and A stands for ambiguous. Tags are set as follows:

- $V = N$ iff $\lambda(\varepsilon) \cap \Sigma_f = \emptyset$: no fault have occurred in ε (even if occurrence of some events in ε_S is uncertain, none of them is faulty),
- $V = F$ iff $\lambda(\varepsilon \setminus \varepsilon_S) \cap \Sigma_f \neq \emptyset$: a fault occurred *before* any of the uncertain events,
- $V = A$ iff $\lambda(\varepsilon \setminus \varepsilon_S) \cap \Sigma_f = \emptyset$ and $\lambda(\varepsilon_S) \cap \Sigma_f \neq \emptyset$: one event in ε is a fault in the uncertain part of ε. There is no guarantee that this event was executed.

A *state estimate* after $w \in \Sigma_o^*$ is a set of possible states $SE = \{(V_i, \varepsilon_S^i)\}$. State estimate SE is said *faulty* if all possible states in SE carry tag F. It is said *normal* if all verdicts in SE are N. Last, it is said *ambiguous* if at least one verdict V_i is equal to A, or there exists two contradictory verdicts $V_i = F$ and $V_j = N$. So, ambiguity appears both when different verdicts originate from different explanations, and when a possible state contains a faulty event which execution is uncertain.

Like summaries, state estimates can be maintained online with observations. Let $SE = \{(V_1, \varepsilon_S^1), \dots (V_k, \varepsilon_S^k)\}$ be the state estimate obtained after observation w,

and let $a \in \Sigma_o$. The update of SE after observation $a \in \Sigma_o$ is the set $SE' = \{(V'_1, \varepsilon_S^{1'}), \dots (V'_1, \varepsilon_S^{k'})\}$ such that for every $(V'_i, \varepsilon_S^{i'})$ of SE' there exists a possible state (V_i, ε_S^i) in SE such that $\varepsilon_S^i \xrightarrow{a} \varepsilon_a$, $\varepsilon_S^{i'} = Sum(\varepsilon_a)$ and:

- $V_i = F$ implies $V'_i = F$,
- $V_i \in \{N, A\}$ and $\lambda(\varepsilon_a) \cap \Sigma_f = \emptyset$ implies $V'_i = N$,
- $V_i \in \{N, A\}$ and $\lambda(\varepsilon_a \setminus \varepsilon_S^{i'}) \cap \Sigma_f \neq \emptyset$ implies $V'_i = F$,
- $V_i \in \{N, A\}$ and $\lambda(\varepsilon_a \setminus \varepsilon_S^{i'}) \cap \Sigma_f = \emptyset$ and $\lambda(\varepsilon_S^{i'}) \cap \Sigma_f \neq \emptyset$ implies $V'_i = A$,

We write $SE \xrightarrow{a} SE'$ when SE' is the state estimate obtained from SE after observation a. One can notice that if ε is an explanation for w, with a frontier F and a summary ε_S, then as ε_S is the part of ε obtained by saturation with unobservable transitions, any process of the form $\varepsilon \setminus \varepsilon_S \cup \varepsilon'_S$ replacing ε_S by an unobservable summary $\varepsilon'_S \in Unfold(\mathcal{N}, \epsilon, F)$ is also an explanation of w. Hence, state estimates should contain all processes that differ only via their summaries, which allows a compact representation (CSE) for state estimates. Let $SE = \{(V_i, \varepsilon_S^i)\}$. SE can be represented as sets of pairs of the form $CSE = \{(V_i, M_i)\}$ where each M_i is the set of places in a frontier F_i appearing in some summary of SE, and V_i is a verdict attached to some summary with frontier F_i. Obviously, one can recover a state estimate from a compact representation that is isomorphic to the original estimate (isomorphism is obtained by relabeling minimal conditions) and diagnosis can safely resume from each M_i for any extension of w, as shown in Proposition 5. A diagnoser that maintains state estimates can claim that an observation corresponds to a faulty (resp. non-faulty) behavior iff all verdicts are set to F (resp. N). A transition relation $CSE \xrightarrow{a} CSE'$ among CSEs can be designed identically to the transition relation among state estimates.

Proposition 6. *Given a safe boundedly silent Petri net $\mathcal{N}$, there exists only a finite number (in $O(2^{3.2^{|P|}})$) of compact representations of state estimates reachable after an observation of an execution of $\mathcal{N}$.*

A *diagnoser automaton* for a safe boundedly silent net $\mathcal{N}$ is an automaton $\mathcal{D}_\mathcal{N} = (S_{\mathcal{D}_\mathcal{N}}, \rightarrow_{\mathcal{D}_\mathcal{N}}, F_{\mathcal{D}_\mathcal{N}})$ such that $S_{\mathcal{D}_\mathcal{N}}$ is the set of reachable compact state estimates for $\mathcal{N}$, $\rightarrow_{\mathcal{D}_\mathcal{N}} \subseteq S_{\mathcal{D}_\mathcal{N}} \times \Sigma_o \times S_{\mathcal{D}_\mathcal{N}}$ is the transition relation among CSEs. $\mathcal{D}_\mathcal{N}$ is deterministic and of size in $O(2^{2^{|P|}})$. It can read observable labels appearing during the execution of a net, and raise an alarm when the reached state estimate is faulty. Note that $\mathcal{D}_\mathcal{N}$ needs not be built a priori to perform diagnosis: one can maintain a single state estimate online, which is updated at each occurrence of a new observation, with a memory in $O(2^{|P|})$. Recall however (Proposition 1) that one needs not build this doubly exponential diagnoser to check diagnosability. However, we will show in the next sections that these state estimators must be used to address costs and to use *disambiguation*.

4 Disambiguation under budget constraints

The construction of section 3 allows to know if a system is diagnosable. The main reason for non-diagnosability is unbounded ambiguity about the actual run executed by a system. However, in safety critical systems, long uncertainty about major failures

is not acceptable. If the actual status of the system is not known and a major failure is suspected, then safety checks must be performed, even if this implies consuming more energy. We model these additional checks as follows: after each observed action, diagnosers have the possibility to perform one test to reduce ambiguity on the actual status of the system. For safe Petri nets, where places can be seen as boolean variables, it seems natural to model tests as mechanisms that provide information of the possible status of places. We first consider whether disambiguation suffices to avoid ambiguity, and then whether it can be performed without exhausting the system's resources.

Definition 10. *A* disambiguation function *is a partial map $d : 2^P \to 2^{2^P}$ that, given an actual marking, returns a set of possible markings the system might be in.*

Disambiguation functions return a set of markings that correspond to a partial observation of an actual (unknown) marking. The knowledge returned by the disambiguation map is not necessarily a state estimation that a diagnoser might have built. So, in general, disambiguation functions provide additional information, that can be used to make a system diagnosable. A first example is map id, that takes a marking M as argument, and returns set $\{M\}$. Another example is the map d_p that associates to every marking M all configurations where $p \in P$ is marked if $M(p) = 1$ and all configurations where p is not marked otherwise. Such a function can be used to inform users on the status of a boolean variable. Note also that the result returned by a disambiguation function needs not be an explicit enumeration of markings, but can be encoded as a constraint on possible markings (e.g. $M(p_1) = 0 \Rightarrow M(p_2) = 1$). In the sequel, we will assume that systems are provided with a finite set $\mathcal{D}is = \{d_1, \ldots, d_n\}$ of (costly) disambiguation functions, and that a diagnoser uses at most one of them after every observation.

Let ε_S be a summary, and let d be a disambiguation function. We say that ε_S is *compatible* with d in marking M if at least one configuration of ε_S is made of places that belongs to $d(M)$. Let ε_S be a summary, d be a disambiguation function, and M be a marking such that ε_S is compatible with d in M. Then, getting information using disambiguation provided by function d can help users refine their estimation of the state of the system. The *refinement* of ε_S by $d(M)$ is denoted by $\varepsilon_{S\setminus d(M)}$, and is the projection of ε_S on successors of configurations of ε_S that are compatible with information provided by $d(M)$. It is computed as follows : $\varepsilon_{S_{\setminus d(M)}} = (E_d, B_d, \lambda_d)$ where $E_d = E \cap \uparrow Cond(\varepsilon_S, d, M)$, $B_d = B \cap \uparrow Cond(\varepsilon_S, d, M)$, with $Cond(\varepsilon_S, d, M) = \{b \in B \mid \exists C \in Conf(\varepsilon_S), b \in C \wedge Place(C) \in d(M)\}$, and λ_d is the restriction of λ to E_d.

Of course, when a system is diagnosable, one does not need disambiguation mechanisms to perform diagnosis. However, if a system is not diagnosable, the questions of diagnosability with the help of tests makes sense. Recall that diagnosis can only be performed after some observation. If ambiguity remains after an observation, then one can perform a test from $\mathcal{D}is$, or wait for a better moment to reduce ambiguity. This situation is well captured as a strategy.

Definition 11. *A* strategy *is a function $\sigma : \Sigma_o^* \to \mathcal{D}is \cup \{d_\perp\}$ that indicates, for every observation w whether a disambiguation from $\mathcal{D}is$ should occur ($\sigma(w) \in \mathcal{D}is$) or if no disambiguation shall be applied ($\sigma(w) = d_\perp$).*

The set of processes compatible with w and pruned by strategy σ is the set of processes built inductively as follows: we first set $\mathbb{E}^0_{w,\sigma} = \{Unfold(m_0, \epsilon, \mathcal{N})\}$ Then,

we compute $\mathbb{E}^{i+1}_{w,\sigma}$ from $\mathbb{E}^{i}_{w,\sigma}$ as follows: we use disambiguation $d = \sigma(w_{[1..i+1]})$ prescribed by σ. Then, for every process $\varepsilon_S \in \mathbb{E}^{i}_{w,\sigma}$, every $\varepsilon' \in Unfold(\varepsilon_S, w_i, \mathcal{N})$ if there exists $M \in Confs(sum(\varepsilon'))$ such that $sum(\varepsilon')$ is compatible wit $d(M)$ we add ε' to $\mathbb{E}^{i+1}_{w,\sigma}$. As application of disambiguation only restricts the set of processes, the notions of summaries, possible states, state estimates and their compact representations (see def. 9) can be used. One can maintain finite sets of state estimates that are compatible with w and pruned by a strategy σ. Similarly, as disambiguation applies after a new observation, a finite set of representations of possible states reached by a net $\mathcal{N}$ together with a verdict can be maintained online with occurrences observable actions with finite memory. **Diagnosability with disambiguation** can now be formalized:

Given a Petri net $\mathcal{N}$ and a set of disambiguation functions $\mathcal{D}is = \{d_1, \ldots, d_n\}$, is there a strategy σ and a constant $K \in \mathbb{N}$ such that for every faulty run ρ of $\mathcal{N}$, and every run $\rho' = \rho.\rho_1$ such that $|\rho'_1|_{\Sigma_o} > K$ every process in $\mathbb{E}^{|w|}_{w,\sigma}$ (i.e. with $w = \Pi_{\Sigma_o}(\rho')$ and pruned by strategy σ) is faulty ?

Theorem 1. *Given a Petri net $\mathcal{N}$ and a set of disambiguation functions $\mathcal{D}is$, one can decide in 2-EXPTIME whether $\mathcal{N}$ is diagnosable with the help of $\mathcal{D}is$.*

Proof (Sketch). We can define diagnosis with disambiguation as a partial information co-Büchi game. We first build an arena which nodes contain: the actual marking of the net, the fault status ($\{N, F\}$) of the system, and state estimates. The game is turn-based. Nodes of this arena are either nodes of the diagnoser, or nodes of the system. From its node, a diagnoser can choose one disambiguation function to refine its state estimation. From its nodes, the system can choose any sequence of action containing an single observation, and hence move to another node of the diagnoser. The partial information in the game comes from the fact that the real state of the system is not known. In this game, the system wins if it can remain ambiguous forever, that is if it can produce a fault, and forces the system to remain in nodes with ambiguous state estimates. The complexity comes from the doubly exponential size of the arena.□

The translation to a partial information co-Büchi game is interesting for reasons to go beyond the simple complexity characterization. A partial information co-Büchi game on an arena G can be translated to a perfect information parity game over an arena G^K that represents knowledge of players, and as parity games are positional [12], it means that bounded memory strategies are sufficient to make a system diagnosable with disambiguation. Usually, and additional exponential cost has to be paid to solve partial information games [6]. However, state estimates already contain knowledge of players. An additional exponential blowup needs not be paid, and solving the game is "only" doubly exponential. At first sight, diagnosis with disambiguation looks close to the diagnosis with adaptive observable alphabet proposed by [5], and to the active diagnosis (see for instance [13]) where some transitions can be disabled to make a system diagnosable. Adaptive observations and disambiguation are orthogonal techniques, and our framework can not be considered as active diagnosis as it does not modify the overall behavior of the monitored system. We discuss these issues in conclusion and in Appendix I.

Information on the current state of a system can be obtained by running some tests, activating sensors, checking for the status of some component, etc. Such operations

usually have a cost, for instance in terms of time or energy. Fortunately, knowing precisely at every instant the current state of a system is not needed to make a system diagnosable. So, one does not have to perform the most expensive tests at every step to ensure diagnosability. In the rest of the paper, we assume that tests follow our energy consumption scheme, and consume energy. We hence extend the cost function to consider the cost of tests. From now, we will consider that a cost functions is a map $\mathcal{C} : T \cup \mathcal{D}is \rightarrow \mathbb{N}$ that associates integers to transitions of T and negative integers to elements of $\mathcal{D}is$. As tests cost energy, they can not be used systematically to disambiguate systems states. The next question to consider is whether one can use tests to disambiguate state estimates of a system without exhausting its energy.

The **ULWUB diagnosability** question is defined as follows: Given a non-diagnosable Petri net $\mathcal{N}$, $\mathcal{D}is$, $\mathcal{C}$, B_0 and B_{max}, is there a strategy that makes $\mathcal{N}$ diagnosable while maintaining the energy provision of the system ≥ 0 ? We will assume that a diagnoser does not know precisely the remaining energy provision of the system. Hence, it uses only its belief, i.e., its estimation of the possible state of the system and bounds for the remaining energy. A diagnoser should take the same disambiguation decision for all sates with the same belief. In the rest of the paper, we show that this question can be answered as an energy game, that can again be solved as a particular instance of co-Büchi game. In a setting where tests have a cost, one may have to wait before launching a test that could exhaust the energy budget of the system. Again, the right moment to perform a test can be defined as a strategy.

Theorem 2. *The ULWUB diagnosability is decidable in* 2*-EXPTIME (in the number of places of the net).*

Proof (sketch). We build an arena where nodes are nodes of the system, or nodes of the diagnoser. As for disambiguation without budget constraints, the diagoser can choose a test to disambiguate its state estimates. Nodes contain the current state of the system, its fault status, the remaining energy provision. They also contain state estimates, and for each summary in state estimates, an upper and lower bound for remaining energy. In system nodes, the system can play any sequence of actions with a single observable event. The current state, fault status, and budget are adapted accordingly. The state and budget estimate are also changed according to the possible executions and their costs. From diagnoser nodes, the diagnoser can choose a particular disambiguation function to refine its sate estimate, which decreases the actual energy budget and the upper/lower estimations by the cost of this test. The diagnoser can also choose to avoid tests, and give its turn to the system without refining its state estimate. If either the system or the diagnoser exhaust the energy budget of the system, the reached node is a particular exhaustion sink node that can not be leaved. Then, in this arena, the diagnoser cannot disambiguate faults with energy constraints iff the system has a strategy to remain in ambiguous or exhaustion nodes forever. This is again a partial knowledge co-Büchi game over a doubly exponential size arena. As for disambiguation, building the knowledge of players in our arena amounts to building copies of state and budget estimates. Hence, an additional exponential blowup needs not be paid, and solving the game is doubly exponential.□

5 Conclusion

Related Work: We have proposed a Petri net based framework for fault detection with disambiguation mechanisms and associated a cost model to this framework. This work is not the first one addressing diagnosis with *Petri nets* (see the survey in [16]). . An offline algorithm to build explanations for an observation using Petri net unfoldings is proposed in [10]. This work targets complete reconstruction of all runs explaining an observation, and provides modular algorithms. Fault detection is a simpler problem than explanation reconstruction, that can be addressed online: the exact run leading to a marking needs not be remembered. A diagnosis framework for general Petri nets is proposed in [1]. The authors exhibit examples of unbounded Petri nets that are diagnosable, i.e., in which every fault can be detected within a finite number of steps even if no general upper bound on the number of observations needed before detection exists. In this setting, diagnosers are no longer regular. [11] represents systems as general Petri nets, and diagnosers as Petri nets too. Diagnoser nets maintain sets of *extended markings*, i.e. reachable markings corresponding to an observation extended with finite vectors of fault status. Aligning extended markings forms a matrix, and occurrence of a fault can be claimed when the column corresponding to a particular fault contains only non-zero entries. With some conditions on the considered net, diagnosis can be performed in a modular way. Extended markings are close to our state estimates, and their construction is close to our notion of observation guided unfolding. However, the approach of [11] enumerates all possible markings (we use a partial order representation that can be obtained efficiently with unfolding techniques -see Appendix D-), and does not address diagnosability.

Diagnosis is often presented as a *passive* process: diagnosers observe a system and emit verdicts. In our setting, diagnosers also have to take decisions to run tests, that affect the energy provision of the monitored system. Our approach can hence be compared with *active diagnosis* settings. In active diagnosis, diagnosers have the possibility to guide the system that they observe. They play the role of a controller that prevents actions to avoid ambiguity in otherwise non-diagnosable systems. The active diagnosability problem then consists in deciding whether such a controller exists. It is EXPTIME-complete for automata [13]. In [2], active diagnosis is recast in a probabilistic setting. The main objective is then to ensure *safe active diagnosability*, i.e. show existence of controller that guarantees diagnosability while ensuring that non-faulty runs still have a strictly positive probability. Active probabilistic diagnosability is EXPTIME-complete, but safe active diagnosis is undecidable. Our approach slightly differs from the active diagnosis approach: active diagnosis forbids some controllable transitions to recover diagnosability, which changes the overall behavior of the system. In our setting, disambiguation functions bring additional information to diagnosers, but using them does not change the behavior of the disambiguated system: for every run ρ of the arena over actions in $\Sigma \uplus \mathcal{D}is$, $\Pi_\Sigma(\rho)$ is a run of the original net $\mathcal{N}$. The *dynamic masks* proposed in [5] are also a form of active diagnosis. In this setting, the observable alphabet can be changed dynamically to ensure diagnosability. This notion of dynamic diagnosability however differs from ours, as we do not change the set of observable actions but rather introduce new actions to refine state estimation. We show in appendix I examples of systems that are diagnosable with dynamic masks and not with disambiguation, and conversely. Hence, both approaches are orthogonal.

Costs have been considered in former works on diagnosis. In [18], a cost is computed for each state of a finite transition system. It represents the effort needed to diagnose a fault. Non-diagnosable systems have diverging costs, and a system is diagnosable if one can associate a finite cost to each state. The cost model considered assigns a weight to each observable action, and the cost associated to a state is a sum of discounted costs of observations for runs leaving this state with a particular dynamic observation strategy (that may change the observable alphabet after each observation). Similarly, Cassez et al. [5] define the cost of diagnosis in terms of weights attached to sets of observable actions. The diagnosers and observers considered are dynamic: the set of observed actions may change depending on the observation. Cost of diagnosis is then the maximal mean cost needed to perform the observations required by the diagnoser. [5] show that checking existence of a k-diagnoser with an optimal cost can be solved in $O(|\Sigma| \times m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$, where n is the number of states and m the number of transitions in the considered automaton. Our framework for diagnosis with costs can be seen as some a quantitative game, as defined in [3]. However, in addition to quantitative considerations, in diagnosis players have to meet/prevent ω-regular objectives. We believe that the setting proposed in this paper easily adapts to most of the cost constraints appearing in [3], such as strict upper and lower bounds on energy level.

Open Issues: Using pieces of processes is a compact way to represent possible states of a system. As shown in Appendix D, summaries can be computed efficiently with unfolding techniques. However, in the worst cases, one may still need to enumerate all possible markings of a net. We believe that in practice this situation does not occur for systems with concurrency, but this has to be demonstrated on case studies, to measure the efficiency gain when working with a Petri net $\mathcal{N}$ and its unfolding rather than directly from its LTS $LTS_{\mathcal{N}}$. In a similar way, in our approach processes and summaries are partial order models, but observations are linearizations. An immediate question is whether our setting can be extended to a fully concurrent one where observations are partial orders too. The natural notion of explanation is when an observed order O *embeds* into the ordering on events depicted in a process ε of $\mathcal{N}$. However, in this setting, diagnosers may not work with finite memory. A possible extension of this work is then to find sensible restrictions on nets that allow diagnosers that only need to memorize summaries of processes and observations of bounded sizes.

Diagnosability with disambiguation and with ULWUB constraints are solved as partial information co-Büchi games in 2-EXPTIME. As for active diagnosis [2, 13] partial observation leads to an exponential blowup in that does not have to be paid twice when solving games. A future work is to find lower bounds. We also plan to consider the impact of several hypotheses of this work on the complexity of ULWUB diagnosability. We have considered that a diagnoser has no information on the remaining energy provision, and has to estimate it, which impacts the size of the arena for the associated co-Büchi game. Now, one can imagine other scenarios: if the diagnoser has access to the remaining energy provision of the system, exhaustion state is not needed, and one does not have to maintain energy estimation. This however should not change the overall complexity of the ULWUB diagnosability game, as the number of moves for the diagnoser keeps the same order of magnitude.

References

1. F. Basile. Overview of fault diagnosis methods based on petri net models. In *Proc. of IEEE European Control Conference (ECC)*, 2014.
2. N. Bertrand, E. Fabre, S. Haar, S. Haddad, and L. Hélouët. Active diagnosis for probabilistic systems. In *Proc. Of FOSSACS 2014*, volume 8412 of *LNCS*, pages 29–42, 2014.
3. P. Bouyer, U. Fahrenberg, K.G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008*, volume 5215 of *LNCS*, pages 33–47, 2008.
4. M.P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu. Diagnosability analysis of unbounded petri nets. In *Proc. of the 48th IEEE Conference on Decision and Control, CDC 2009*, pages 1267–1272, 2009.
5. Franck Cassez and Stavros Tripakis. Fault diagnosis with static and dynamic observers. *Fundam. Inform.*, 88(4):497–540, 2008.
6. K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *Proc. of Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17*, volume 6397 of *LNCS*, pages 1–14, 2010.
7. K. Chatterjee, L. Doyen, T.A. Henzinger, and J-F Raskin. Algorithms for omega-regular games with imperfect information. In *Proc. Of Computer Science Logic, 20th International Workshop, CSL 2006*, volume 4207 of *LNCS*, pages 287–302, 2006.
8. K. Chatterjee, L. Doyen, T.A. Henzinger, and J-F Raskin. Algorithms for omega-regular games with imperfect information. *CoRR*, abs/0706.2619, 2007.
9. J. Esparza, S. Römer, and W. Vogler. An improvement of Mc Millan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
10. E. Fabre, A. Benveniste, S. Haar, and C. Jard. Distributed monitoring of concurrent and asynchronous systems*. *Discrete Event Dynamic Systems*, 15(1):33–84, 2005.
11. S. Genc and S. Lafortune. Distributed diagnosis of discrete-event systems using petri nets. In *Applications and Theory of Petri Nets (ICATPN) 2003*, volume 2679 of *LNCS*, pages 316–336, 2003.
12. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*, 2002.
13. S. Haar, S. Haddad, T. Melliti, and S. Schwoon. Optimal constructions for active diagnosis. In *Proceedings of FSTTCS'13*, volume 24 of *Leibniz International Proceedings in Informatics*, pages 527–539, 2013.
14. T. Jéron, H. Marchand, S. Pinchinat, and M-O. Cordier. Supervision patterns in discrete event systems diagnosis. In *Workshop on Discrete Event Systems, WODES'06*, pages 262–268, 2006.
15. S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.
16. M. Pocci. *Test and diagnosis of discrete event systems using Petri nets*. PhD thesis, Univ. Aix-Marseille & Univ. Cagliari, 2013.
17. M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.
18. D. Thorsley and D. Teneketzis. Diagnosis of cyclic discrete-event systems using active acquisition of information. In *WODES'06, 8th INternational Workshop on Discrete Events Systems*, pages 248–255, 2006.

A Proof of Proposition 1

Proposition 1 Deciding whether a Petri net $\mathcal{N}$ is diagnosable is $PSPACE$-complete. Furthermore, this can be decided in $O(2^{2.|P|})$.

Proof. We first show that diagnosability is in PSPACE. Without loss of generality, we assume that considered nets are live, i.e. they do not deadlock. Within this setting, a system is not diagnosable iff there exists a pair of infinite runs ρ, ρ' where ρ is faulty, ρ' is non-faulty, and their observations are identical. Hence, to show that a system is non-diagnosable, it suffices to prove that such a pair of runs exists. Notice that the number of reachable markings for a safe net is bounded by $2^{|P|}$. To find such runs, one can build a machine that remembers:

- a boolean b indicating whether a fault has occurred.
- An integer n counting the length of runs
- A pair of markings M_1, M_1'. M_1 represents a marking reached after a faulty run ρ_1 of $\mathcal{N}$, and M_1' the current marking of an exploration that continues after ρ_1
- A pair of markings M_2, M_2'. M_2 is a marking reached after a non-faulty run ρ_2 that has the same observation as ρ_1. M_2' is the current marking of a non faulty exploration that continues after ρ_2.

Then one can proceed in three phases: the first phase consists in searching for a pair M_1, M_2, where M_1 is a marking reached after a faulty run ρ_1 of $\mathcal{N}$, and M_2 is a marking reached after a non-faulty run ρ_2 that has the same observation as ρ_1. Obvously, using a simple pumping argument, one only needs to consider runs with less than $2^{|P|}$, observable actions. The second phase searches for a pair of markings accessible from the markings reached during the first phase, and that can be the beginning of an infinite part of the pair of executions we are trying to build. The last phase shows that the guess of the second part was correct by exhibiting a loop. One can proceed as follows:
Phase 1: The algorithm starts with $n = 0$, $M_1 = M_2 = m_0$. At each step of the algorithm, one randomly selects an observable letter a from Σ_o, and searches a run ρ_1 starting from M_1 that has a as unique observable action (this can obviously be done in PSPACE, with the help of an additional counter n' bounded by $2^{|P|}$, and of temporary markings). If no run can be found, then the algorithm fails. If not, then one has found a run with observation a starting from M_1 that is either faulty or not, and ends in a new marking M_1'. If a fault was encoutered, then bit b is set to 1. Similarly, one can search in PSPACE a non-faulty run strating from M_2 of length $< 2^{|P|}$ with observation a. If such run is not found then the algorithm fails. Otherwise, we increase the number of observed steps, i.e. $n = n + 1$. If $n > 2^{|P|}$ and $b = 0$, then one has explored a sufficiently long path without finding a fault and the algorithm stops. If $n < 2^{|P|}$ and $b = 0$ then the first phase continues from $M_1 = M'1$ and $M_2 = M_2'$. If $n < 2^{|P|}$ and $b = 1$ then a pair of compatible faulty and non faulty runs was found, and we can proceed to the second phase.
Phase 2: The second phase consists in selecting a pair of markings that are accessible from M_1, M_2 via runs with the same observation (the run from M_2 should remain non-fautly), and are a guess of a strating point for a cyclic behavior. One selects an integer k smaller than $2^{|P|}$ and tries to extend executions from M_1 and M_2 with k observations,

while ensuring that the execution starting from M_2 remains non-faulty, and that both executions remain equivalent. This phase ends with a pair of markings M_1, M_2 that are guessed to be the origins of a pair of cyclic executions with identical observations.

Phase 3: The third phase consists in finding a pair of executions that return to M_1, M_2 with the same observable sequence of events, and with less than $2^{|P|}$ steps. The algorithm proceeds by random exploration as before (ensuring that the second run is still non-faulty). It has to memorize M_1, M_2 and maintain a pair of current markings M_1', M_2'. It stops after $2^{|P|}$ steps or if the pair of current markings returns to M_1, M_2. In this case, one has witnessed a pair of runs of the form $\rho = \rho_1.\rho_1'$ and $\rho' = \rho_2.\rho_2'$ that are respectively faulty and non-faulty, and such that $\rho_1.\rho_1'^k$ and $\rho_2.\rho_2'^k$ have identical observation for any value of k, hence showing that $\mathcal{N}$ is not diagnosable.

Overall, this non-deterministic algorithm uses a bit of information, four markings, and all explorations can be done with a pair of counters bounded by $2^{|P|}$. The algorithm is hence in NPSPACE, and by Savitch's lemma it is in $PSPACE$.

Hardness comes from a reduction from the vacuity of language intersection problem. Assume a pair of languages $L_1, L_2 \subseteq \Sigma^*$ given as a pair of automata $\mathcal{A}_1 = (Q_1, \rightarrow_1, q_0^1, Q_F^1)$, $\mathcal{A}_2 = (Q_2, \rightarrow_2, q_0^2, Q_F^2)$. One can build a safe Petri net $\mathcal{N}_{\mathcal{A}_1,\mathcal{A}_2} = (P, T, \lambda)$ with $P = Q_1 \cup Q_2 \cup \{p_0\} \cup \{p_F\}$ labeled by letters from $\Sigma \cup \{u, f, a\}$ where $a \notin \Sigma$ is an observable letter, f is a fault and u is an unobservable action. Every action in Σ is observable. $T = t_u \cup t_f \cup t_F \cup \{t_{q,\sigma,q'} \mid (q, \sigma, q') \in \rightarrow_1 \cup \rightarrow_2\} \cup \{t_{F,i} \mid q_{F,i} \in Q_F^1 \cup Q_F^2\}$. We set $\lambda(t_{q,\sigma,q'}) = \sigma$, ${}^\bullet(t_{q,\sigma,q'}) = q$, $({}^\bullet t_{q,\sigma,q'}) = q'$. We set $\lambda(t_f) = f$, $\lambda(t_u) = u$, where f is a fault, and u is unobservable, and ${}^\bullet t_f = {}^\bullet t_u = \{p_0\}$, ${t_f}^\bullet = q_0^1$, ${t_u}^\bullet = q_0^2$. For each transition $t_{F,i}$, ${}^\bullet t_{F,i} = q_{F,i}$ ${t_{F,i}}^\bullet = p_F$, and we set $\lambda(t_{F,i}) = a$ where a is a letter that does not appear in Σ. Last we have ${}^\bullet t_F = {t_F}^\bullet = \{p_F\}$ and $\lambda(t_F) = a$. An illustration of this construction is provided in Figure 4. Then, every faulty run of $\mathcal{N}$ is a run that starts with f, and produces an observation $w.a^k$ such that $w \in L_1$ and every non-faulty run is a run that starts with u, and produces an observation $w'.a^{k'}$ with $w' \in L_2$. Clearly, there is an ambigous observation in $\mathcal{N}$ iff $L_1 \cap L_2 \neq \emptyset$.

We can now prove that diagnosis can be performed in $O(2^{2.|P|})$. As $\mathcal{N}$ is safe, one can compute a transition system that accepts all runs of $\mathcal{N}$ and which states of the form (b, M) remember a boolean value b indicating whether a fault has occurred or not and a marking of $\mathcal{N}$. This LTS has at most $2^{|P|+1}$ states. Then, one can use a twin machine construction [15]. This twin machine is a synchronous product of an automaton with a fault-free version of the same specification. The two automata synchronize on observable transitions. This machine can be used to check existence of infinite run over pairs of states that comport one faulty component and another non-faulty one (see also [5] for this construction). It is well known that a regular system is diagnosable iff such runs do not exist in its twin machine. As the twin machine is of quadratic size, this means that diagnosability of $\mathcal{N}$ is in $O(2^{2.|P|})$. □

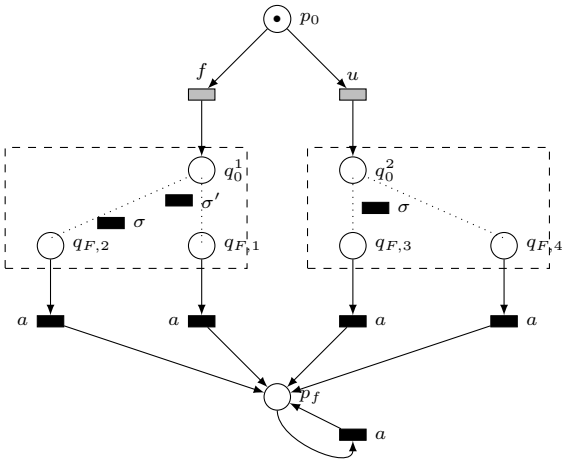

Fig. 4. Construction of a net $\mathcal{N}_{\mathcal{A}_1,\mathcal{A}_2}$ from two automata.

B proof of Proposition 2

Proposition 2:: The ULWUB problem is in PSPACE (w.r.t. the number of places in $\mathcal{N}$).

Proof. The ULWUB problem was considered for weighted automata in [3]. For a weighted automaton, the problem is decidable in P. The decision algorithm consists in detecting path of the form $\rho_1.(\rho_2)^\omega$ called *lassos*, i.e. where $W_{C\downarrow B}(\rho_1.\rho_2) > W_{C\downarrow B}(\rho_1.\rho_2.\rho_2)$ If such a lasso exists, then there is a way for the system to consume more energy than it produces. This detection can be performed in polynomial time using a variant of the Bellman-Ford algorithm. For safe Petri nets, we can obtain a weighted automaton by translating $\mathcal{N}$ to its equivalent LTS $LTS_{\mathcal{N}}$, of size in $O(2^{|P|})$, and then assigning $\mathcal{C}(t)$ as weight for each transition (m, t, m') in $LTS_{\mathcal{N}}$. As Bellman-Ford algorithm's complexity is in $O(m.n)$ for an automaton with m transitions and n vertices, we have an EXPTIME algorithm solving the ULWUB question in $O(2^{2.|P|}.|T|)$.

For Petri nets we use the characterization of behavior leading to exhaustion proposed by [3], i.e. detect paths of length $< 2^{|P|}$ that end with a negative energy provision, or lassos that end with a cycle of negative cumulated weight. These lassos are paths of the form $\rho_1.\rho_2$, where ρ_1 is an acyclic path, and ρ_2 a cyclic path such that $\sum_{t\in\rho_2} \mathcal{C}(t) < 0$. That is, at every iteration of the cycle decreases the energy provision of the system. We can easily show that is it sufficient to consider path of bounded size. Suppose a path ρ of size $> 2^{|P|}$ ending with a negative energy budget. Then, this path can be decomposed into $\rho = \rho_1\rho_2\rho_3$, where ρ_2 is a cycle. If ρ_2 has a positive weight, then $\rho_1.\rho_3$ is a smaller path that also exhausts the system's energy. If ρ_2 has a negative weight, then $\rho_1.\rho_2$ is a lasso and can be captured by lasso search part. Now assume that one has detected a lasso of the form $\rho_1.\rho_2$, where ρ_2 is a cycle and a negative cost $-k$. If ρ_2 is of size greater that $2^{|P|}$, then it can be decomposed into $\rho_{21}.\rho_{22}.\rho_{23}$ where ρ_{22} is a cycle. If the cost of ρ_{22} is positive, then $\rho_{21}.\rho_{23}$ is a cycle of cost smaller than ρ_2. If the cost of ρ_{22} is negative,

then $\rho_1.\rho_{21}.\rho_{22}$ is also a lasso of size smaller than ρ. If ρ_1 is of size greater than $2^{|P|}$, then it can be decomposed into $\rho_1 = \rho_{11}.\rho_{12}.\rho_{13}$ where ρ_{12} is a cyclic part. If the cost of ρ_{12} is positive, then one can reach ρ_2 with a lower energy budget that with ρ_1, and if it is negative, then $\rho_{11}.\rho_{12}$ is a lasso of smaller size that exhausts the energy budget of the system. Hence, one can run nondeterministic exploration algorithms to find path of size smaller that $2^{|P|}$ that exhaust the system, or lassos which cyclic part is of size smaller than $2^{|P|}$ and which cyclic part is of size smaller than $2^{|P|}$ and has a negative cost. Maintaining the energy budget remaining, the cost of a cycle, and the length of explored path can be done with space in $O(|P|)$. Again, by Savitch's lemma, we can claim that the ULWUB problem is in PSPACE.□

C Proof of Proposition 3

Proposition 3: Let $\mathcal{N}$ be a boundedly silent Petri net, with bound K, and let $w \in \Sigma_o^*$ be an observation. Then, $\mathbb{E}_w$ is finite, and contains processes built in at most $(|w|+1).K$ steps.

Proof. We proceed by induction on the length of observations.

Let us first consider the base case $w = \epsilon$. The inductive construction of $\mathbb{E}_w$ starts with $B_0 = \{(\perp, p) \mid p \in M_0\}$, $E_0 = \emptyset$. As $\mathcal{N}$ is boundedly silent, we can add at most K unobservable transitions to $\varepsilon_0 = (B_0, E_0, \lambda_0)$. If $m < K$ unobservable transitions are successfully added to obtain a process ε_m, then the projection of any linearization of ε_m on observable actions is an empty word, and hence, ε_m is an explanation of w. Once can also notice that every extension of a process with an observable transition results in an order with at least one event, hence with a linearization that differs from $w = \epsilon$. As we know that $\mathcal{N}$ is boundedly silent, every process built with more than K transitions contains an observable action and hence is not an explanation of ϵ. Let us denote by IX_w the set of processes built inductively to explain word w, and in particular IX_ϵ the set of processes built inductively starting from m_0 that contains all unobservable processes of $\mathcal{N}$. Obviously, we have $|\varepsilon| < K$ for every $\varepsilon \in IX_w$.

We can not proceed with the induction phase. Let $w \in \Sigma_o^*$ be a word of size n, let $IX_w = \mathbb{E}_w$ be the sound and complete set of explanations built inductively for w, satisfying

- $|\varepsilon| < (|w|+1).K$ for every $\varepsilon \in IX_w$
- $IX_w = \mathbb{E}_w$

Let $w' = w.a$, with $a \in \Sigma_o$. We can build $IX_{w'}$ inductively from IX_w, i.e. starting from each process $\varepsilon = (E_N, B_n, \lambda_n)$ of IX_w, try to append a event representing an occurrence of a transition t with $\lambda(t) = a$). If ε can not be extended with an observable transition, then this process can not be a prefix of an explanation for w'. Conversely, let us suppose than one can append an event $e = (X, t)$ with $\lambda(t) = a$ to ε to obtain a new process ε_a. Then either e causally depends from events in E_n, or it is completely independent. In both cases, the partial order O_{ε_a} has a linearization of the form $w.a$, and is hence an explanation of w'. Similarly, one can saturate ε_a with occurrences of unobservable transitions, and obtain a process which has $w.a$ as linearization. This

saturation process stops after at most $K-1$ steps, as $\mathcal{N}$ is boundedly silent. Hence, $IX_{w'}$ is finite, and contains processes obtained in at most $(|w|+1).K+K$ steps, i.e in $(|w'|+1).K$ steps.

It now remains to show that $IX_{w'} = \mathbb{E}_{w'}$. Obviously, we have $IX_{w'} \subseteq \mathbb{E}_{w'}$, and one only builds processes which are explanations for w'. Now let us suppose that there exists a process $\varepsilon \in \mathbb{E}_{w'}$ that is not an element of $IX_{w'}$. Let $e_a = (X,t)$ be a maximal event such that $\lambda(t) = a$, i.e. for which all causal successor are unobservable events, and let $\varepsilon' = \varepsilon \setminus \uparrow e_a$. We have that ε' is an explanation of w, and is hence contained in $IX_w = \mathbb{E}_w$. Furthermore, conditions ${}^\bullet e_a$ are maximal in ε'. Hence ε' can be extended to obtain ε and ε belongs to $IX_{w'}$, raising a contradiction.□

D An algorithm to compute $\mathbb{E}_w$

Let $w \in \Sigma_o^*$ be the observation of some run of $\mathcal{N}$. We can build inductively a set $\mathbb{E}_w = \{\varepsilon_w^i\}$ of processes that "explain" w as follows. At each step, we build a set of explanations $\mathbb{E}_w^{n+1}$ from $\mathbb{E}_w^n$. Each process ε_n^i in $\mathbb{E}_w^n$ explains a prefix of w. If ε_n^i can not be extended with a new transition, and does not provide an explanation for w, it is simply forgotten. If the process can not be extended, but already contains an explanation for w, it appears in $\mathbb{E}_w^{n+1}$ (and will be kept in the fix point $\mathbb{E}_w$). If one can find a way to extend ε_n^i with new transitions while still explaining some prefix of w, then for every transition one creates a new process in $\mathbb{E}_w^{n+1}$. The construction stops when no process of $\mathbb{E}_w^n$ can be extended.

Every process $\varepsilon_w^i = (E^i, B^i, \lambda^i) \in \mathbb{E}_w$ is hence built inductively. At some steps of the inductive construction, some processes are dropped when they cannot be extended to explain w. Every process ε_w^i is composed of a set of events E^i, a set of conditions B^i, $\lambda^i : E \to \Sigma$ and built inductively as follows. We start from $\varepsilon_w^0 = \varepsilon^0$ with $\varepsilon^0 = (E_0^0, B_0^0, \lambda_0^0)$ with $B_0^0 = \{(\bot, p) \mid p \in M_0\}$, $E_0^0 = \emptyset$, and λ_0^0 the empty map.

At each step, we can build several processes $\varepsilon_{n+1}^j = (E_{n+1}^j, B_{n+1}^j, \lambda_{n+1}^j)$ from $\varepsilon_n^i = (E_n^i, B_n^i, \lambda_n^i)$ as follows: Let $Max(\varepsilon_n^i) = \{b \in B_n^i \mid b^\bullet = \emptyset\}$ be the set of maximal places in ε_n^i. We also compute $imax(\varepsilon_n^i) = |\{e \in E_n^i, \lambda_n^i(e) \in \Sigma_o\}|$, the index of the last letter of w matched in ε_n^i.

One can notice that $Place(Max(\varepsilon_n^i))$ is exactly the configuration of $\mathcal{N}$ after executing transitions appearing in ε_n^i in an order that is compatible with the observation up to $imax(\varepsilon_n^i)$. From $Max(\varepsilon_n^i)$, we can compute a set of firable transitions $Firable(\varepsilon_n^i) = \{t \in T \mid {}^\bullet t \subseteq Place(Max(\varepsilon_n^i))\}$. As the main goal when building processes guided by an observation w is to find all possible explanations for w, in particular faulty (and unobservable) ones, we will extend ε_n^i with transitions that are either non observable, or are labeled by the next letter in w, i.e. letter of index $imax(\varepsilon_n^i)+1$. A transition t *can be appended* to ε_n^i if either $\lambda_{\mathcal{N}}(t) \in \Sigma_{uo}$, or $imax(\varepsilon_n^i) < |w|$,$\lambda_{\mathcal{N}}(t) \in \Sigma_o$ and $\lambda_{\mathcal{N}}(t) = w_{[imax(\varepsilon_n^i)+1]}$. We can then choose one of the appendable transitions t of ε_n^i and add it to ε_n^i to build a process $\varepsilon_{n+1}^j = (E_{n+1}^j, B_{n+1}^j, \lambda_{n+1}^j)$ where:

- $E_{n+1}^j = E_n^i \uplus \{e = (X,t)\}$ with $X = Max(\varepsilon_n^i) \cap Place^{-1}({}^\bullet t)$
- $B_{n+1}^j = B_n^i \uplus \{(e,p) \mid p \in Place(t^\bullet)\}$

– $\lambda^j_{n+1}(x) = \begin{cases} \lambda^i_n(x) \text{ if } x \in E^i_n \\ \lambda_{\mathcal{N}}(t) \text{ if } x = (X,t) \notin E^i_n \end{cases}$

We can create one new processes ε^j_{n+1} from ε^i_n for each appendable transition of ε^i_n. As w is an observation of $\mathcal{N}$, $\mathbb{E}_w$ contains at least one process. Due to concurrency and choices, several transitions can usually be appended to a process. Hence the observation guided construction above is non-deterministic. Every unobservable transition can be appended to ε^i_n if it is allowed from configuration $Max(\varepsilon^i_n)$, and all observable transitions must carry label $w_{[imax(\varepsilon^i_n)+1]}$, but there can be several occurrences of such transitions in a labeled net. As the set of appendable transitions may contain conflicting transitions $\mathbb{E}_w$ can contain more than one process.

For each process, the construction stops when no transition can be appended to the current process ε^i_n. This corresponds to two situations: on one hand, if ε^i_n is not a complete explanation of w, then the unfolding is blocked, because the transitions that were formerly chosen to build ε^i_n lead to a configuration from which observation w can not be exhibited. The second situation is when a complete explanation for w is found, process ε^i_n can not be extended with additional observable transitions without inserting new letters that do not appear in w, and no more unobservable transition can be fired either. The process obtained in the first case is not a member of $\mathbb{E}_w$, and the process obtained in the second case is a member of $\mathbb{E}_w$.

The algorithm 1 below computes the set of processes that explain an observation w. Processes are built incrementally by appending silent transitions, or a transition carrying the next letter to explain. Note that by doing so, one may explore the whole set of markings of a net.

Algorithm 1: $Unfold(\mathcal{N}, w, m_0)$: unfolding of $\mathcal{N}$ w.r.t. observation $w \in \Sigma_o^*$ starting from m_0

input : A net $\mathcal{N}$, A word $w \in \Sigma_o^*$
output : A set of processes $\mathbb{E}_w$
$B_0 = \{(\perp, p) \mid p \in m_0\}$;
$X = \{E_0 = (\emptyset, B_0, \lambda_0)\}$;
$Blocked = \emptyset$;
$Explanations = \emptyset$;
while $X \neq \emptyset$ **do**
  choose a process $\varepsilon = (E, B, \lambda)$ in X;
  $X = X \setminus \{\varepsilon\}$;
  Compute $Firable(\varepsilon)$;
  nbextensions =0;
  for $t \in Firable(\varepsilon)$ **do**
    if $\lambda_{cN}(t) \in \Sigma_{uo}$ **then**
      $\varepsilon' = (\ E \cup \{e = (X, t)\}, B \cup (e, t^\bullet),$
        $\lambda \cup e \to \lambda_{\mathcal{N}}(t))$;
      $X := X \cup \varepsilon'$;
      nbextensions++;
    end
    if $\lambda_{cN}(t) = w_{[imax(\varepsilon)+1]}$ **then**
      $\varepsilon' = (\ E \cup \{e = (X, t)\}, B \cup (e, t^\bullet),$
        $\lambda \cup e \to \lambda_{\mathcal{N}}(t))$;
      $X := X \cup \varepsilon'$;
      nbextensions++;
    end
  end
  if *nbextensions ==0* **then**
    if ε *explains* w **then** $Explanations = Explanations \cup \varepsilon$;
    else $Blocked = Blocked \cup \varepsilon$;
  end
end
return(Explanations);

A possible improvement for algorithm 1 is to rely on the notion of branching processes, that factorize common parts of processes. A branching process of a net $\mathcal{N}$ can be seen as "processes with conflicts". Conflict free subsets of these branching processes are exactly processes of $\mathcal{N}$. Construction of Branching processes have been well studied (see for instance [9]). However, in our setting, we have an additional difficulty with respect to a simple unfolding: in order to provide only explanations of an observation, labeled transitions of a branching process have to be ordered as in the observed word. It can be cumbersome to maintain a branching process together with a progression of each branch in the explanation of an observation. We give an informal algorithm 2 below, inspired from the unfolding algorithm of Esparza [9], that builds a branching process from a marking for a *single observed letter*. In this algorithm, we consider given

the procedure $PE(BP)$ that computes a set of *possible events* to add to the branching process. As for processes, possible events are pairs of the form (X, t). However, in branching processes, X is not necessarily a subset of maximal conditions, but it is a *co-set*, i.e. a set of conditions that have no conflicting predecessors. In addition, (X, t) must not already appear in the branching process. It is known that projection of branching processes on conflict-free subsets are processes of the unfolded net. It is also known that this construction is efficient: branching processes allow to compute processes of a net without exploring all its markings.

The algorithm proceeds as follows: we assume a dummy condition $(\perp, \sigma)$ that is required by any transition t with label σ. That is, if $(X, t) \in PE(BP)$ and $\lambda(t) = \sigma$, we add $(\perp, \sigma)$ to the co-set X allowing t. Then we unfold the net using only unobservable transitions, and at most one occurrence of σ in each process. As condition $(\perp, \sigma)$ is used only once, this guarantees that any process in our branching process contains at most one occurrence of transition labeled σ. The unfolded net is boundedly silent, so this construction stops (one can not build infinite processes with only one occurrence of σ and a finite number of silent transitions).

Algorithm 2: $Unfold(\mathcal{N}, \sigma, m_0)$: unfolding of $\mathcal{N}$ w.r.t. letter $\sigma \in \Sigma_o$ starting from m_0

input : A marking m_0, a letter $\sigma \in \Sigma_o$
output : A set of processes $\mathbb{E}_\sigma$
$B_0 = \{(\perp, p) \mid p \in M_0\} \cup (\perp, a)$;
$BP = (\emptyset, B_0, \lambda_0)\}$;
Compute $PE(BP)$
while $PE \neq \emptyset$ **do**
    choose t in $PE(BP)$;
    if $\lambda(t) = \epsilon$ *or* $\lambda(t) = \sigma$ **then**
        add $e = (X, t)$ to BP;
        Compute $PE(BP)$;
    end
end
return conflict free sets of BP that contain a transition labeled σ;

Obviously, algorithm 2 can be adapted to saturate an existing process with unobservable transitions.

E Properties of frontiers and silent parts of processes

Proposition 7. *$Frontier(\varepsilon)$ is a marking of ε.*

Proof. We proceed by induction on the number of observable events in the process. For the base case, consider a process ε without observable event, starting from marking M. Then $\varepsilon_S = \varepsilon$ and the frontier of ε is M. Now consider a process ε, with a silent part ε_S and a frontier M, that is a marking of ε. There is hence an execution ρ starting from M,

and leading to execution of the whole silent part. Now, consider an extension of ε with an event $e = (X, t)$. If $\lambda(t) \in \Sigma_{uo}$, then the silent of $\varepsilon \cup \{e\}$ is simply $\varepsilon_S \cup \{e\}$, and its set of minimal places M remain unchanged. If $\lambda(t) \in \Sigma_o$, then the silent part is composed of successor conditions of e, and of elements of ε_S that are not predecessors of e.Hence, there exists an execution $\rho'.\rho_\epsilon$ where $\rho' = M \xrightarrow{t_1} M_1 \dots \xrightarrow{t} M_e$ of $\varepsilon \cup \{e\}$ starting from M that starts with all predecessors of e, then executes e, and ends with an execution ρ_ϵ of remaining silent transitions in ε. The frontier of $\varepsilon \cup \{e\}$ is the marking M_e that contains $e^\bullet$ and all conditions of M that were not consumed during execution of ρ'.□

F Proof of proposition 5

Proposition 4: Let ε be an explanation of observation $w \in \Sigma_o^*$, and ε_S be its summary. Then for every marking M of ε_S, $Place(M)$ is a marking that is reachable via a run ρ of $\mathcal{N}$ such that $\lambda(\rho) = w$.

Proof. The proof is a straightforward of the following observation : M is a marking that is reachable in ε_S from $Frontier(\varepsilon)$, and $Frontier(\varepsilon)$ is a reachable marking of $\mathcal{N}$ via an execution which observation is w.□

proposition5: Let ε be an explanation of word $w \in \Sigma_o^*$, and M be its frontier. Then $\varepsilon' \sqsupseteq \varepsilon$ is an explanation of $w.a \in \Sigma_o^*$ iff there exists $\varepsilon_a \in Unfold(\mathcal{N}, a, M)$ such that $\varepsilon' = \varepsilon \cup \varepsilon_a$ (where union of processes means union component wise).

Proof. Let ε be a process of $\mathcal{N}$, an let M be its frontier. Let $\varepsilon_a \in Unfold(\mathcal{N}, a, M)$. If $\varepsilon \cup \varepsilon_a$ is a process (i.e. it does not contain pairs of events with non empty intersection of their presets), then it is a process of $\mathcal{N}$ that explains $w.a$, as it continues unfolding of ε wrt w by addition of new unobservable events, adds a transition labeled by a from places of ε, and does not create conflicts. Note that union does not duplicate unobservable event that already appeared in ε, as the way events are defined depend only on their preset, and hence $e \cup e = e$. Then, $\varepsilon \cup \varepsilon a \sqsupseteq \varepsilon$, and it is a process that explains $w.a$. Now, suppose that $\varepsilon' \sqsupseteq \varepsilon$ is an explanation for $w.a$. As ε explains w, the part of ε' that contains a transition labeled by a is contained in $\varepsilon' \setminus \varepsilon$. Furthermore, this transition is the only observable one in $\varepsilon' \setminus \varepsilon$. It now remains to show that $\varepsilon'' = (\varepsilon' \setminus \downarrow M \cup M \in Unfold(\mathcal{N}, a, M)$. We already have that Let us suppose that ε' is a causally closed subset of elements in ε', with minimal elements equal to M. We also know that the only observable transition in ε'' is labeled by a. As ε' is a process ε'' is also conflict free. We hence have that $\varepsilon'' \in Unfold(\mathcal{N}, a, M)$.□

Proposition 6 Given a safe boundedly silent Petri net $\mathcal{N}$, there exists only a finite number (more precisely at most $2^{3.2^{|P|}}$) of compact representations of state estimates reachable after an observation of an execution of $\mathcal{N}$.

Proof. For boundedly silent PNs, compact representations of state estimates are subsets of tagged markings of $\mathcal{N}$ representing a finite set of finite potential unobservable summaries of processes with a status of the system. As $\mathcal{N}$ is safe, the number of such markings is at most $2^{|P|}$, and the number of state estimates is at most $2^{3.2^{|P|}}$. Note that the number of state estimates is in $O(2^{2^{|P|+2}})$ hence is doubly exponential in $|P|$. □

G Proof of Theorem 1

Theorem 1 Given a Petri net $\mathcal{N}$ and a set of disambiguation functions $\mathcal{D}is$, one can decide in 2-EXPTIME whether $\mathcal{N}$ is diagnosable with the help of $\mathcal{D}is$.

Proof. The question can be brought back to a co-Büchi game with partial observation. We build an arena $G = (Q, \longrightarrow, q_{init})$, with two players 0 (the diagnoser) and 1 (the system). The set of nodes Q is partitioned in $Q_0 \uplus Q_1$, where Q_0 depicts nodes of the diagnoser and Q_1 nodes of the system. Nodes are tuples of the form $q = (M, ES, Tag)$, where M is a marking, and Tag is a element from $\{F, N\}$ denoting whether the actual run of the system ending in M is faulty or non faulty, ES is a state estimate. From its nodes, the diagnoser can choose to use a disambiguation function $d_i \in \mathcal{D}is$ and update its state estimation to $ES_{\setminus d_i(M)}$ or keep his knowledge of the system as it is (i.e., use function $d_\perp$ that brings no information on the current state of the system). From nodes of Q_1 the system performs sequences of actions that result in a single observation. A move in the arena is a triple of the form (q_i, d, q_j), where $q_i \in Q_0, q_j \in Q_1$ and $d \in \mathcal{D}is \cup d_\perp$, or (q_i, a, q_j), where $q_i \in Q_1, q_j \in Q_0$ and $a \in \Sigma_o$ (the game is turn-based). Moves from player 1 are labeled by observations, and moves from player 0 by disambiguation functions. A node (M, ES, Tag) is *ambiguous* if its state estimate component ES is ambiguous, and faulty if $Tag = F$. We denote by $Amb(G)$ ambiguous nodes and by $Faulty(G)$ faulty nodes of G.

We build Q and moves inductively as follows: we first define the initial node $q_{init} = (m_0, ES_0 = Unfold(m_0, \epsilon, \mathcal{N}), T_0 = N) \in Q_1$, i.e. we start from the initial marking and an estimation of states that may have been reached without producing an observation. Obviously, the system starts in a normal situation, whence the tag $T_0 = N$. We also assume that initial state of the system is precisely known from the diagnoser, hence application of disambiguation by player 0 is not needed immediately from the start, so the initial node q_{init} belongs to player 1.

Moves for player 0 are defined as follows: (q_i, d, q_j) is a move from player 0 iff $q_i \in Q_0$, d is a disambiguation function, q_i is a node of the form $q_i = (M, ES = \{(V_1, \varepsilon_1), \ldots, (V_k, \varepsilon_k)\}, T)$ and $q_j = (M, ES_{\setminus d(M)}, T) \in Q_1$, where $ES_{\setminus d(M)}$ is the state estimate (suffixes) that can be obtained from ES after application of disambiguation with the information provided by $d(M)$. Note that T does not change, as the current status of the system is not modified by application of a disambiguation: a faulty system remains faulty when the diagnoser gets more knowledge on its actual state. In particular, we systematically have moves of the form $(q_i, d_\perp, q_j)$, where $q_i = (M, ES, T) \in Q_0$ and $q_j = (M, ES, T) \in Q_1$. Hence, Q_0 and Q_1 may contain copies of similar nodes.

Moves for player 1 are defined as follows: (q_i, a, q_j) is a move from player 1 iff $q_i \in Q_1, a \in \Sigma_o$, q_i is a node of the form $q_i = (M, ES = \{(V_1, \varepsilon_1), \ldots, (V_k, \varepsilon_k)\}, T) \in Q_1$ and $q_j = (0, M', ES', T') \in Q_0$, where M' is a marking such that $M \stackrel{a}{\Longrightarrow} M'$, ES' is the state estimate such that $ES \stackrel{a}{\longrightarrow} ES'$, and T' can have the following values: it must be set to F if $T = F$, of if all runs from M to M' labeled by a are faulty runs (i.e. contain a faulty transition). It must be set to N if $T = N$ and no run from M to M' labeled by a is faulty. Last, if $T = N$, and runs from M to M' labeled by a can be either safe or faulty, then both moves from q_i to $(M', ES', N) \in Q_0$ and $(M', ES', F) \in Q_0$ appear in the arena.

This construction terminates, and builds an arena of size at most in $O(2^{2^{|P|}})$, i.e. the size of the arena is mainly influenced by the size of state estimates. We set as observation function the map that returns the state estimate for a given $q \in Q$. We set a co-Büchi objective $Win = Amb(G) \cap Faulty(G)$ for player 1. We recall that an infinite run is winning for a co-Büchi objective if the set of states visited infinitely often is contained in Win. Then, we claim that the net $\mathcal{N}$ can be disambiguated with the help of $\mathcal{D}is$ iff player 1 has no winning strategy for the Co-Büchi game with arena G and objective Win, that is player 1 wins if it can force the system to execute infinite faulty ambiguous runs (that never visit a non-ambiguous node).

Let $inf(\rho)$ be the nodes visited infinitely often in an infinite run ρ of the arena. Remember that disambiguation does not change the behavior of the system, and hence can not prevent faults occurrences, and cannot prevent player 1 to reach $Faulty(G)$. Suppose that there exists a strategy σ_1 such that for every disambiguation strategy σ_0, runs of the form $\rho.(\rho_1^\omega)$ compatible with strategies σ_0, σ_1 where ρ is faulty satisfy $inf(\rho_1^\omega) \subseteq Win$. Then, $\rho.\rho_1$ contains a fault, and ρ_1 contains only faulty ambiguous states. So, after every run of the form $\rho.(\rho_1)^k$, the state estimation is ambiguous for any value of k, and $\mathcal{N}$ is not diagnosable. Conversely, suppose that there exists a strategy σ_0 such that for every strategy σ_1 of player 1, every run guided by σ_1 is of the form $\rho.(\rho_1^\omega)$ with $inf(\rho_1^\omega) \not\subseteq Win$. Then, ρ_1 contains at least one non-ambiguous state. Then, if ρ or ρ_1 is a faulty run, the fault is detected. So, in every cyclic behavior of $\mathcal{N}$ occurring after a fault, there is at least one state estimation than is not ambiguous, and the system is diagnosable. Co-Büchi games with partial observation are EXPTIME-complete [6]. However, the EXPTIME blowup comes from the computation of an arena that contains beliefs of players. In our setting, the only unknown information is the actual current marking of the system and its status. Hence, belief computation will not result in a new exponential blowup, and will simply compute copies of state estimates. Then, finding whether disambiguation can help a diagnoser is in 2-EXPTIME. □

H Proof of Theorem 2

Theorem 2: The ULWUB diagnosability is decidable in 2-EXPTIME (in the number of places of the net).

Proof. As for diagnosis with disambiguation, we build an arena $G = (Q, \longrightarrow, q_{init})$ over a set of nodes $Q = Q_0 \uplus Q_1$, where Q_0 depicts the nodes of the diagnoser and Q_1 the nodes of the system. Each node of the arena is a tuple of the form $(M, ES, T, val, L, U, Lconf, Uconf)$ where M is the actual marking of the system, ES is a state estimate, T is a tag, val is the remaining energy provision, L and U are respectively lower and upper bound estimations of this remaining provision according to observation, and $Lconf, Uconf$ are maps that associate the lowest possible provision to the minimal configuration in possible states of ES.

As we are working with integer provisions between 0 and B_{max}, we can build an arena remembering every provision in $0 \dots B_{max}$. From every node of player 0, one can play a test $d_i \in \mathcal{D}is$. If the cost of d_i is smaller or equal to the actual energy provision in the node, the state estimation is disambiguated. We model this as a refinement of state

estimate, and compute new values for $val, L, U, Lconf, Uconf$. Otherwise, the energy level of the system is exhausted, and we model this situation with a move to a sink losing node $\perp \in Q_1$. Then the diagnosability question boils down to deciding if there exists a winning strategy for player 0 to prevent infinite ambiguous cycles and avoid sink node $\perp$.

We first set $q_{init} = (m_0, ES_0 = Unfold(m_0, \epsilon, \mathcal{N}), T_0 = N, val_0 = B_0, L_0 = B_0, U_0 = B_0, Lconf_0 = Uconf_0) \in Q_1$, where $Lconf_0$ associates value B_0 to m_0: the system starts from its initial marking, with its initial energy budget B_0, and has performed no fault yet. As this state is equivalent from the point of view of the diagnoser to any other state that is reachable via silent transitions, the state estimate is $Unfold(m_0, \epsilon, \mathcal{N})$. As no observation was produced, the diagnoser does not have to emit a verdict nor disambiguate the system, so $n_0 \in Q_1$.

Then, there exists a move (q_i, d, q_j) iff $q_i \in Q_0$ and is a node of the form $q_i = (M, ES, T_i, val_i, L_i, U_i, Lconf_i, Uconf_i)$, $val_i + \mathcal{C}(d) > 0$, and $q_j \in Q_1$ is a node of the form $q_j = (M, ES', T_j, val_j, L_j, U_j, Lconf_j, Uconf_j)$, where: $ES' = ES_{\setminus d(M)}$, $T_j = T_i$, $val_j = val_i + \mathcal{C}(d)$. We also have $L_j = L_i + \mathcal{C}(d)$ and $U_j = U_i + \mathcal{C}d$. Similarly, $Lconf_j$ associates $Lconf_i(M_i) + \mathcal{C}(d)$, and $Uconf_j$ associates $Uconf_i(M_i) + \mathcal{C}(d)$ to every minimal configuration of a possible state of ES that is compatible with d. Both maps are of course restricted to minimal configurations in possible states of ES'. In particular, we have moves of the form $(q_0, d_\perp, q_1) \in Q_0 \times \{d_\perp\} \times Q_1$ with $q_0 = q_1 = (M, ES, T, val, U, L, Lconf, Uconf)$ indicating that the diagnoser choses to use no disambiguation and waits for the next observation.

Let us now consider moves of the system (moves from Q_1), and how they update the energy provision. Remember provisions are needed to ensure that a chosen strategy does not exhaust the system's energy. The only way for the system to restore its energy provision is to play sequences of actions which cost is positive. The safe strategy for a diagnoser is to consider that upon occurrence of an observable action a, the energy provision has changed by the minimal possible value since the last observation. This smallest value and the possible status of the system after observing a can be computed as follows.

Let M be a marking, a be an action, T be a status in N, F, and $val \in \mathbb{N}$. Then one can compute the minimal budget needed to reach marking M' with status T' while observing a. This is done by a standard exploration of paths that contain exactly one observable action a. We build successive configurations, and remember at each step the current marking, whether a fault has occurred, whether action a was observed, and the remaining energy provision. We start from $R^0 = (M, T, b_a^0 = ff, val)$, and we successively compute $R^{i+1} = R^i \cup \bigcup_{r \in R^i} post_a(r)$ where $(M', T', b'_a, v') \in post_a((M, T, b_a, v))$ iff $\exists t, M[t\rangle M'$, $v' = \max(v + \mathcal{C}(t), B_{max})$, and either $\lambda(t) \in \Sigma_f$, $T' = F$, and $b'_a = b_a$, or $\lambda(t) = a$, $T' = T$, $b_a = ff$ and $b'_a = tt$, or $\lambda(t) \in \Sigma_{uo} \setminus \Sigma_f$, $T' = T$, $b'_a = b_a$. Transitions with $\lambda(t) \in \Sigma_o \setminus \{a\}$ and with $\lambda(t) = a$ when $b_a = tt$ are forbidden. As $\mathcal{N}$ is boundedly silent, the construction ends with a fixpoint R. Then, one can compute $Min_R(M, T, a, val) = \{(M_i, T_i, tt, v_i) \mid \forall (M_i, Ti, tt, v'), v' \geq v\}$. For every entry (M_i, T_i, tt, v_i) in $Min_R(M, a, val)$, v_i is the smallest energy provision remaining when reaching marking M_i and tag T_i from marking M with tag T and energy provision val

while observing a. Similarly, one can compute $Max_R(M, T, a, val)$ the highest energy provision remaining when going from M to another marking while observing a.

We now define system moves. There is a move (q_i, a, q_j) iff $q_i \in Q_1$ and is a node of the form $q_i = (M, ES, T, val_i, L_i, U_i, Lconf_i, Uconf_i)$, $q_j \in Q_0$ and is a node of the form $q_j = (m', ES', T', val_j, L_j, U_j, Lconf_j, Uconf_j)$ such that $(m', T', tt, val_j) \in Min_R(m, T, a, val_i)$, ES' is the state estimate such that $ES \xrightarrow{a} ES'$, and $val_j > 0$. We can now compute L_j and U_j. For a state estimate $ES = \{(V1, \varepsilon_1), \ldots, (V_k, \varepsilon_k)\}$, we define $Min(ES)$ as the set of minimal configurations in each process. $L_j = \min\{val \mid \exists M_i \in Min(ES), \exists (M'_i, tt, val) \in Min_R(M_i, a, Lconf(M_i))\}$ and $U_j = \max\{val \mid \exists M_i \in Min(ES), \exists (M'_i, tt, v) \in Max_R(M_i, a, Uconf(M_i))\}$. Similarly, we can update $Lconf_i$ and $Uconf_j$ to consider most/less expensive ways to move from $Min(ES)$ to $Min(ES')$ while observing a (details are omitted).

Last, we model moves of the system and of the diagnoser that exhaust the provision of the system. These are moves of the form $(q_i, d, \bot)$, where $q_i = (m, T, ES, val) \in Q_0$ with $val + \mathcal{C}(d) \leq 0$, or of the form $(q_i, a, \bot)$, where $q_i = (m, T, ES, v) \in Q_1$ and such that there exists $(M', T', tt, val) \in Max_R(m, a, T, v)$ with $val \leq 0$. The last kind of move is $(\bot, \epsilon, \bot)$. With this last transition, provision exhaustion forces infinite behaviors looping on node $\bot \in Q_1$.

We keep the same meaning as before for $Amb(G)$ and $Faulty(G)$ (ambiguous/faulty nodes of the arena), and we define $Win = Amb(G) \cap Faulty(G) \cup \{\bot\}$. We claim that the system $\mathcal{N}$ with initial budget B_0 is not diagnosable with disambiguation under ULWUB constraints iff player 1 has a winning strategy in the co-Büchi game made of the arena G and winning condition Win. Again, we can reuse the results of [6] showing that co-Büchi games with imperfect information are EXPTIME-Complete. It was also shown in [8] that there exists a deterministic strategy in parity games of imperfect information iff there exists a strategy in a perfect information parity game over a structure G^K that is exponentially larger, and which nodes represent the beliefs of players. It is well known that parity games are determined and positional. Hence, beliefs are sufficient to have a winning strategy if it exists. However, as for diagnosability with disambiguation, the belief on states and on remaining budget that can be computed from nodes of the arena is already contained in G, and we hence do not suffer this additional exponential blowup. Hence, the ULWUB diagnosability question can be answered on an arena of size in $|G|$. The number of possible markings is in $O(2^{|P|})$, the number of state estimates is in $O(2^{2^{|P|+2}})$, and each state estimate is a subset of markings and tags, of size at most $2^{|P|+2}$. Encoding lower and upper bounds for state estimates can be done by affecting a value in $[1, B_{max}]$ to each state estimate regardless of its contents. It can hence be encoded as a map from $2^{|P|+2}$ to $1..B_{max}$, and there are $B^{2^{|P|+2}} = 2^{2^{|P|+2} \log B}$ such maps. From the above description, G is of size in $O\left(2^{|P|} \times 2^{2^{|P|+2}} \times 2 \times B \times 2^{2.2^{|P|+2} \log B})\right)$. □

I On differences between Masks and disambiguation

The dynamic setting proposed by [5] is the following: after each obsered action, one can choose dynamically which subset of letters from Σ_o (called a *mask*) can be observed.

Note that Σ_o remains unchanged, and one can only choose masks contained in Σ_o. The main idea is to avoid using costly observations when only a subset of them suffices to guarantee diagnosability. The Figure 5 below shows a system with 6 states, and alphabet $\Sigma = \{f, a, b, c\}$, where f is a fault. For the system represented in Figure 5, if $\Sigma_o = \{a, b, c\}$, then choosing to observe actions $\{b, c\}$ after observation a is sufficient to make the system diagnosable. However, if c can not be observed (i.e. $\Sigma_o = \{a, b\}$), then the system is not diagnosable with a mask technique, because observations will be words of the form $a.b^k$ that main correspond to executions wit or without faults. On the contrary, consider the net at the bottom of Figure **??**. Its associated LTS is isomorphic to the automaton on top of the Figure. Equipping the system with a disambiguation function that allows to know the status of place p_{bad}, or p_2 suffices to make the system diagnosable, even if b, c are unobservable. Now, if disambiguation functions in $\mathcal{D}is$ only give information on the status of p_0, p_1, p_3 and c is unobservable, then the system is not diagnosable with the help of $\mathcal{D}is$.

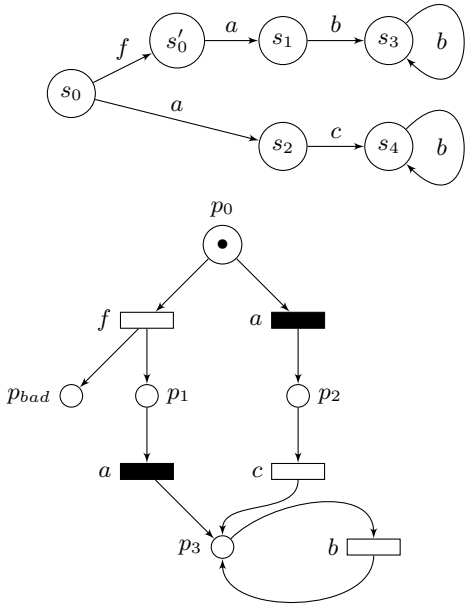

Fig. 5. An example that shows difference between disambiguation and observation masks