Learning and Analytics in Intelligent Systems 39

George A. Tsihrintzis
Maria Virvou
Haris Doukas
Lakhmi C. Jain *Editors*

Advances in Artificial Intelligence-Empowered Decision Support Systems

Papers in Honour of Professor John Psarras

Springer

George A. Tsihrintzis · Maria Virvou ·
Haris Doukas · Lakhmi C. Jain
Editors

Advances in Artificial Intelligence-Empowered Decision Support Systems

Papers in Honour of Professor John Psarras

Springer

Editors
George A. Tsihrintzis
Department of Informatics
University of Piraeus
Piraeus, Greece

Maria Virvou
Department of Informatics
University of Piraeus
Piraeus, Greece

Haris Doukas
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece

Lakhmi C. Jain
KES International
Shoreham-by-Sea, UK

ISSN 2662-3447 ISSN 2662-3455 (electronic)
Learning and Analytics in Intelligent Systems
ISBN 978-3-031-62315-8 ISBN 978-3-031-62316-5 (eBook)
https://doi.org/10.1007/978-3-031-62316-5

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Chapter 14
Supporting Decision-Making in Diagnosis of Discrete-Event Systems by Model-Based Temporal Techniques

Gianfranco Lamperti, Stefano Trerotola, and Marina Zanella

Abstract Decision support is essential when humans are responsible for choosing critical courses of action about large, complex, distributed, partially-observable, dynamical systems. When the functioning of the system looks abnormal, a decision is expected to be taken based on the root cause of the undesired behavior. Still, several alternative root causes, or *candidates*, each consisting of a set of *faults*, can explain the same observation(s). Since finding such candidates requires heavy diagnostic reasoning, the literature describes a vast collection of automated diagnosis tools. When the diagnosis tool is *model-based*, the knowledge stored in the system model is drawn not only from human experts but also from design and/or operation data. Up to a few years ago, model-based diagnosis was *set-oriented*, a candidate being a set of faults that accounts for the observation(s). Recently, a *temporal-oriented* perspective to diagnosis of dynamical systems was proposed: a candidate has become a chronological *sequence* of faults, and the set of all candidates has turned into a regular language over the alphabet whose symbols are faults. This new perspective may help a human operator to better understand what actually took place inside the system, thus supporting the decision-making process more adequately. This chapter deals with the decision support provided by a model-based temporal-oriented approach to diagnosis of partially-observable discrete-event systems. A (distributed) discrete-event system consists of components that are modeled as communicating automatons. Three temporal-oriented diagnosis techniques, which differ for the amount of compiled knowledge they manage (if any), are investigated: *interpreted* diagnosis, *compiled*

G. Lamperti (✉) · S. Trerotola · M. Zanella
Department of Information Engineering, Via Branze 38, University of Brescia, 25123 Brescia, Italy
e-mail: gianfranco.lamperti@unibs.it

S. Trerotola
e-mail: stefano.trerotola@outlook.it

M. Zanella
e-mail: marina.zanella@unibs.it

G. A. Tsihrintzis et al. (eds.), *Advances in Artificial Intelligence-Empowered Decision Support Systems*, Learning and Analytics in Intelligent Systems 39,
https://doi.org/10.1007/978-3-031-62316-5_14

diagnosis, and *hybrid* diagnosis. Experimental results suggest that both interpreted and compiled diagnoses suffer from serious complexity difficulties, while hybrid diagnosis may be applicable in real contexts.

Keywords Decision support systems · Model-based diagnosis · Temporal reasoning · Discrete-event systems · Active systems · Automatons · Knowledge compilation

14.1 Introduction

The expression *decision-support system* (DSS) has a broad meaning as it generically represents a computerized program that can help human users come to judgments or make more informed actionable decisions at a quicker pace. This definition is independent of the specific application domain, of the goal(s) of the supported judgments and decisions, as well as of the technology embedded in each software program considered. A DSS typically provides and (possibly) compares alternative solutions relevant to a given problem instance and/or predicts the consequences of different decisions that the user can make. The range of domains where a DSS can be adopted is increasingly wide: it may be helpful in organizations (e.g. to achieve smart manufacturing [21]), business (e.g. to explore different policies about product sales [12, 13]), agriculture (e.g. to predict the areas where a natural substance that controls pests will be effective [18]), and healthcare (e.g. to choose a treatment plan for a patient or to increase the impact of public programs [31]). Further target areas and tasks of interest for existing DSSs are listed in the introductory section of [24]. A whole collection of application fields that have been addressed by DSSs is surveyed in [22], where all encompassed DSSs share the same technology, namely multi-agent systems. Different (AI as well as non-AI) technologies incorporated in distinct approaches are instead surveyed in [3], where all the DSSs considered share the domain of psychiatry. Several state-of-the-art DSSs are qualified as *intelligent* inasmuch they are based on AI techniques, including neural networks, genetic algorithms, multi-agent systems, fuzzy models, machine learning, and hybrid systems. As remarked in [22], an intelligent DSS is highly recommended to treat complex problems in dynamical and distributed environments: these environments are the focus of the current chapter.

Finding a diagnosis is a complex problem that can traditionally benefit from DSSs, both in medical [3, 36] and industrial [30, 44] areas. The present chapter copes with the task of finding a diagnosis, specifically, by facing the problem of providing a decision support to humans who are responsible for critical decisions about the behavior of large, complex, distributed, partially-observable, dynamical systems, either natural or artificial. In this context, a decision is expected to be made when the system behavior looks abnormal, possibly coming to an abrupt stop. Finding the root causes, namely the *faults*, of the undesired observed behavior requires diagnostic reasoning, which is typically hard and may bring to several alternative *explanations*,

an explanation consisting in one or more faults that altogether can have produced the observed behavior. A tool that is able to compute precisely and efficiently such kind of explanations, possibly based on preference criteria, is undisputedly helpful for decision making. This is why in the last decades the AI community has proposed several techniques for carrying out the diagnosis task for either static or dynamical systems. For most of them, model-based technology is adopted, where a behavioral system model is processed by a domain-agnostic engine in order to infer the alternative explanations, each of which is a *candidate*. Hence, a model-based diagnosis tool is a knowledge-based tool that computes the candidates that can explain the given observation of a given system. The knowledge, which is possibly drawn not only from human experts but also from design and/or operation data, is stored in the system model. The dynamical systems considered by the approach described in this chapter are a category of asynchronous distributed discrete-event systems (DESs) called *active systems* in the literature [27].

In the past, model-based diagnosis of DESs was typically employed in processing alarms in operations rooms, both for electricity [16, 17] and telecommunication networks [34]. Power transmission devices and autonomous unmanned helicopters are the focus of attention in [16, 26], respectively. A more recent challenge for model-based diagnosis of DESs is represented by the huge realm of cyber-physical systems [15].

Up to a few years ago, model-based diagnosis of both static and dynamical systems was invariably set-oriented: a candidate was a *set* of faults that accounts for a given observation. In 2020, the notion of a chronological *sequence* of faults, namely a *temporal fault*, was introduced [4–7]. This notion is adopted in the present chapter, since we believe that it can better support decision making. However, computing temporal-oriented candidates is harder than computing set-oriented ones; hence, the execution time performance of three distinct techniques to accomplish temporal-oriented diagnosis of DESs have been evaluated empirically.

In the remainder of the chapter, a background about the diagnosis task in the literature is provided in Sect. 14.2. The concept of a (distributed) DES is recalled in Sect. 14.3. Section 14.4 presents a temporal-oriented approach to (a posteriori) diagnosis in three variants: interpreted, compiled, and hybrid. Section 14.5 outlines the implementation of a DSS, and shows the results of the relevant experimental activity. Section 14.6 draws some conclusions.

14.2 Background and Motivation

A diagnosis task hypothesizes what took place inside a system or process, given a collection of (outward) observations. Different hypotheses can be suggested for the same given observations owing to the partial observability of the system or process considered. For instance, in medical diagnosis, a bunch of symptoms is the observed behavior, while the corresponding diagnostic hypotheses are the possible causes (internal to the human body) that can account for these symptoms. In AI, the

challenge of automating the diagnosis task was first faced by diagnostic *rule-based* systems, such as *MYCIN* [41], which, starting from the second half of the 1980s, were replaced by *model-based* diagnosis systems [19, 23, 38]. The latter find the causes of the observed behavior of the considered (natural or artificial) system by exploiting its (behavioral) model. The system to be diagnosed usually consists of several *components*, hence its model is *distributed*, that is, it defines explicitly the inner behavior of each component, whereas the behavior of the system as a whole is implicit. In fact, at least in theory, the global system behavior is derivable from the component models and the interconnections between components.

The alternative hypotheses relevant to a group of observations are called *candidates*, then a *set* of candidates is the ordinary output of the diagnosis task. Each candidate produced by *consistency-based diagnosis* has no logical inconsistency with the observed behavior, while each candidate computed by *abduction-based diagnosis* entails the observed behavior. Both forms of artificial reasoning in model-based diagnosis can adopt *strong fault models*; otherwise, consistency-based diagnosis can adopt *weak fault models*. When the component models are weak, they specify only the normal behavior; hence, the diagnosis task can hypothesize which components are not working properly, namely the *faulty* components. If the models are strong, they specify a behavior that encompasses both normality and abnormality; hence, a (possibly null) specific *fault* can be hypothesized for each component. It is assumed that a component (and, therefore, a system too) can be affected by faults that belong to a finite set. A specific misbehavior in a component or in the system as a whole is caused by either one or several faults.

Each component model maps the input(s) to the output(s). In a *static* system, e.g. a combinational circuit, this mapping is time-independent. Some input pins and some output pins are relevant to the static system as a whole: the system input values and the system output values altogether form the observed behavior processed by the diagnosis task.

In a *dynamical* system, e.g. a sequential circuit, the mapping is time-dependent, hence model-based diagnosis has to find out the system state also [43]. For some purposes, a dynamical system is advantageously modeled as a *discrete-event system* (DES) [11], which changes its state over (qualitative) time. According to this abstraction, the time tags of state changes are disregarded, while only their chronological order is retained. A (possibly unobservable) input value drives each state change. Several representation formalisms can be adopted for a DES, however: typically, a (single) finite-state machine [29], a Petri net [1, 2, 10, 14, 20, 28, 37, 45], a network of synchronous finite-automatons (FAs), like in [39, 40], or a network of asynchronous FAs, like in this chapter. Each FA models the behavior of a distinct component. These models are untimed: only the reciprocal order of state transitions is reckoned to be interesting, while no time length is explicitly considered. Notwithstanding a remarkable exception [35], the fault models usually adopted for DESs are strong, as is the case with the synchronous DESs in a seminal work [39, 40], and with active systems [27], which are considered in the present chapter. Each state transition is either free of faults, namely *normal*, or affected by a specific fault, namely it is *faulty*. Distinct faulty transitions may share the same fault. While a DES

is being operated, a chronological sequence of events is perceived, where each single perception is called an *observation*: altogether such sequence is called a *temporal observation*. A *trajectory* of a DES is a sequence of state transitions, then it generates a temporal observation. The diagnosis task computes candidates that correspond to all (and only) the trajectories that produce (i.e. entail) the given temporal observation, which means that DES diagnosis can be regarded as a form of *abductive* reasoning [32].

If a given temporal observation is regarded as a monolithic input, and only one diagnosis output is produced in order to account for it altogether, the task is dubbed *a posteriori* diagnosis. This kind of diagnosis is the focus of the present chapter. If, instead, single observations or observation chunks (a chunk consisting in some consecutive observations) are progressively provided, so as to trail the system while it is running, and a diagnosis output is produced for each of them, the task is called *monitoring-based* diagnosis. A posteriori diagnosis hypothesizes what has happened inside the system over the span when the temporal observation was originated. This task is carried out, for instance, in order to uncover the causes of an (abrupt or even catastrophic) halt in a system. The output of a posteriori diagnosis is a single set of candidates; they can help a diagnostician make a decision, possibly involving repair actions. In monitoring-based diagnosis, instead, the set of candidates is updated continuously as the task is typically performed when the system is monitored, by sampling sensor readings and periodically sending them to a control unit. The outputs of monitoring-based diagnosis is useful to feedback the evolution of the system as well as to perform maintenance actions.

Irrespective of the variety of models considered by distinct approaches and of the task being executed either a posteriori or during monitoring, before 2020, model-based diagnosis of both static and dynamical systems was invariably *set-oriented*. In static systems, a candidate is a set of either faulty components or specific faults assigned to components. Likewise, in DESs, a candidate is a *set* of all the faults associated with the abnormal state transitions taking place in a trajectory of the DES that generates the temporal observation. Every single candidate and the whole set of candidates are finite and bounded as the domain of faults is finite. While set-orientation looks like appropriate for static systems, it sounds odd for DESs since a candidate is the projection of a chronologically ordered *sequence* of state transitions. Faults, which are possibly causally chained and/or intermittent, occur sequentially in the real world. Unfortunately, within a set, the pieces of information about the chronological order of faults and their repetitions are totally lost, since a set does not include any duplicates and its elements are unordered.

Every time a new observation chunk is processed, monitoring-based diagnosis enables one to discover whether some new faults have occurred; however, it does not enable one to ascertain whether there has been any repetition of a fault that has already manifested itself. In other words, not even monitoring-based diagnosis can help the diagnostician in acquiring awareness of the repetitions of *intermittent* faults.

Hence, a new *temporal-oriented* viewpoint on DES diagnosis was proposed [4–7], according to which a candidate, instead of being a set of faults, is a sequence of

faults. This sequence, called *temporal fault*, is relevant to a trajectory that produces a (given) temporal observation, thus it is possibly unbounded.

A temporal fault is different from a usual candidate for three main reasons: (1) it consists in the *bag* of faults that took place in the trajectory, which contains each and every instance of the same fault, (2) it clearly shows the chronological order of the fault occurrences, and (3) its length may be unbounded as (intermittent) faults may occur cyclically, and each cycle may be unobservable. The temporal-oriented diagnosis output consists in all distinct temporal faults, which may be infinite in number. Each distinct temporal fault is pertaining to a set of DES trajectories, each of which produces the (finite) temporal observation; the cardinality of such set of trajectories is possibly infinite (as a trajectory may include a cycle of normal unobservable state transitions).

The extra pieces of information contained in temporal faults may be essential for helping diagnosticians make decisions in critical scenarios. Our claim is that a temporal-oriented approach provides more adequate support to decision making as it can help an operator better understand what actually took place inside the system: it does not merely highlight the faults that are possibly affecting the DES, like set-oriented diagnosis does, it also places these faults within a chronologically ordered line. Hence, on the one hand, a temporal-oriented perspective is meant to provide a better cognitive support to a human diagnostician in charge of deciding what to do next, by offering a more realistic view of what has happened; on the other, the possible infinite cardinality of the set of candidates makes it difficult to display this output to a human operator. Providentially, an (even infinite) set of candidates that account for a temporal observation is a regular language over the alphabet whose symbols are faults. Thus, the temporal-oriented diagnosis output of a DES becomes a regular expression: each temporal fault is indeed a string matching this expression.

The content of a regular expression is more significant and useful than that of a collection of sets of faults. Let us assume that the set-oriented output relevant to a given DES diagnosis problem instance is $\{\{u, v\}, \{v, w\}\}$, while the temporal-oriented output is $v(uuu \mid w)v$. The former informs that a pair of faults is affecting the DES, either $\{u, v\}$ or $\{v, w\}$. This implies that fault v has taken place for sure since it is included in both candidates. A diagnostician may wonder whether fault v was the first to occur, this case having a quite different meaning and severity. This piece of information, which is missing in the set-oriented output, is instead provided by the temporal-oriented diagnosis, according to which fault v occurred first; so now the diagnostician can appreciate whether the situation is severe or not. In addition, according to the temporal-oriented output, fault v has surely manifested itself twice, and, between the pair of v's, there were either three u's or one w. The repetition of v is possibly meaningful for an expert, e.g. for it exonerates some evolutions of the considered system, the same as the faults in between. For instance, it may be easy to unearth whether sequence vwv has presented itself, hence the diagnostician will look into it, thus discarding one alternative and catching the real candidate. Identifying the real candidate is instead more complex and time consuming when the diagnosis output is set-oriented only.

Since the temporal-oriented diagnosis output is richer than the set-oriented diagnosis output, a couple of questions are in order: (1) *which algorithms can compute the former*, and (2) *which are the performances of these algorithms?* A recent work [25] has addressed these questions within the context of *monitoring-based* diagnosis of DESs. The following sections address the same questions for *a posteriori* diagnosis of DESs. To this end, three techniques are presented to reply question (1) above, while experimental evidence is provided to answer question (2).

14.3 Modeling of DESs

A distributed DES is a net of interconnected components, the behavior of each of which is modeled by means of a communicating automaton [8]. The interconnections are called links: they are channels in which events are sent. A component may change its state by a transition; each transition is triggered by an event. An event is external if it occurs outside of the DES, internal otherwise. An internal event comes from another component and it is conveyed by a link. After the triggering event has been consumed, other events are possibly generated on links, thus triggering new component transitions in a chain reaction.

Example 1 Figure 14.1 shows a physical device, involving two components, a *protection* and a *breaker*, that is aimed at controlling a line in an electrical grid. The device is designed in such a way that, when the line is affected by a short circuit (e.g. caused by a lightning), which is detected by a drop in voltage, the protection sends an open command to the breaker so as to eliminate the short circuit. When, the short circuit has vanished and the voltage has become normal, the protection sends a close command to the breaker in order to bring back the electrical power. The DES modeling this device, called $\mathcal{W}$ (*watcher*), is represented on the left of the figure: it consists of protection p and breaker b, along with a link from p to b. The communicating automatons of the two components are displayed on the right of the figure. Specifically, there are two states and four transitions in the model of p, and two states and eight transitions in the model of b. As detailed in Table 14.1, a triple

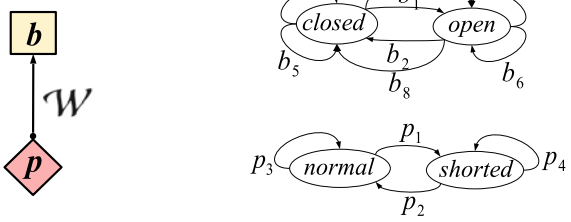

Fig. 14.1 Left: DES watcher $\mathcal{W}$, which is composed of a protection p, a breaker b, and a link from p to b; right: communicating automatons of b (top) and p (bottom)

Table 14.1 Description of the transitions of breaker b and protection p (cf. Fig. 14.1)

Transition	Action
$b_1 = \langle closed, (op, \emptyset), open \rangle$	b receives an open command and opens
$b_2 = \langle open, (cl, \emptyset), closed \rangle$	b receives a close command and closes
$b_3 = \langle closed, (op, \emptyset), closed \rangle$	b receives an open command and (abnormally) remains closed
$b_4 = \langle open, (cl, \emptyset), open \rangle$	b receives a close command and (abnormally) remains open
$b_5 = \langle closed, (cl, \emptyset), closed \rangle$	b receives a close command and remains closed
$b_6 = \langle open, (op, \emptyset), open \rangle$	b receives an open command and remains open
$b_7 = \langle closed, (cl, \emptyset), open \rangle$	b receives a close command and (abnormally) opens
$b_8 = \langle open, (op, \emptyset), closed \rangle$	b receives an open command and (abnormally) closes
$p_1 = \langle normal, (sh, \{op\}), shorted \rangle$	p reacts normally to a short circuit by generating the open command
$p_2 = \langle shorted, (ok, \{cl\}), normal \rangle$	p reacts normally to a short extinction by generating the close command
$p_3 = \langle normal, (sh, \{cl\}), normal \rangle$	p reacts abnormally to a short circuit by generating the close command
$p_4 = \langle shorted, (ok, \{op\}), shorted \rangle$	p reacts abnormally to a short extinction by generating the open command

$\langle x, (e, E), x' \rangle$ denotes a component transition from a state x to a state x', triggered by an input event e and generating a set of output events E.

It is assumed that only one component transition can occur at a time. The DES moves from its initial state to a *quiescent* state, that is, a state where there are no events in the links, by following a series of component transitions, called a *trajectory* of the DES. The DES state changes upon each (component) transition in a trajectory. A *trajectory segment* is a subsequence of contiguous (component) transitions in a trajectory. Even if it is possibly infinite, the set of all the trajectories of the DES can be represented as a deterministic FA (DFA), namely the *space* of the DES.

Definition 1 The *space* of a DES $\mathcal{Y}$ is a DFA,

$$Space(\mathcal{Y}) = \left(\Sigma, X, \tau, x_0, X_q\right), \tag{14.1}$$

where the alphabet Σ is the set of component transitions, X is the set of states, where a state is a pair (C, L), with C being the array of the states of the components and L being the array of the (possibly empty[1]) events within links, τ is the transition function mapping a state and a component transition into a new state, $\tau : X \times \Sigma \mapsto X$, x_0 is the initial state, and X_q is the set of quiescent states.

[1] Formally, an empty link contains an empty event, denoted ε.

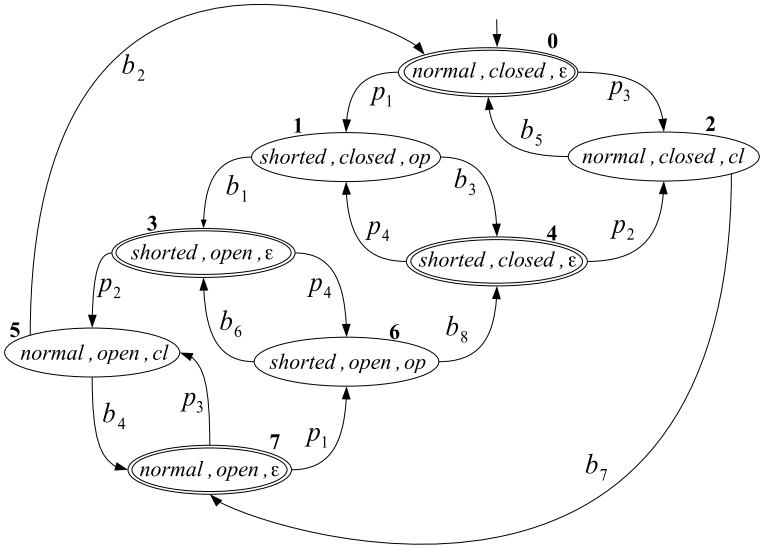

Fig. 14.2 Space of the watcher $\mathcal{W}$ (cf. Fig. 14.1)

Example 2 Figure 14.2 depicts $Space(\mathcal{W})$; each space state is a triple $(\bar{p}, \bar{b}, e)$, with $\bar{p}$ and $\bar{b}$ being the states of p and b, respectively, while e is the event in the link. For easy referencing, the space states are renamed $0 \cdots 7$. The initial state is 0 and 0, 3, 4, and 7 are the quiescent states. Due to cycles in the space, there are infinite trajectories of $\mathcal{W}$, one of them being $T = [p_3, b_5, p_1, b_3, p_4, b_3, p_2, b_5]$.

Since we are interested in diagnosis, the model of a DES $\mathcal{Y}$ is extended with information about the *observability* and the *abnormality* of the DES, which are both specified in a *mask*, namely $Msk(\mathcal{Y})$, which is a set of triples (t, o, f) where, for each component transition in $\mathcal{Y}$, a corresponding (possibly empty) observation o and a (possibly empty) fault f are specified. If $o \neq \varepsilon$, then transition t is *observable*, else t is *unobservable*; also, if $f \neq \varepsilon$, then t is *faulty*, else t is *normal*.

Example 3 Table 14.2 shows the mask of the DES $\mathcal{W}$ (cf. Example 1), where the domain of observations is $\{po, bo\}$ and the domain of faults is $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{h}\}$. Incidentally, just one observation is defined for p and b, namely po and bo, respectively. Notice how transition p_1 is normal and observable, p_3 is faulty and unobservable, while b_7 is faulty and observable. The perception of an observation is insufficient to trace the relevant transition, as such observation is possibly shared by several transitions.

The mask allows us to associate each trajectory with a *temporal observation*.

Definition 2 The *temporal observation* of a trajectory T of a DES $\mathcal{Y}$ is the sequence of the observations associated with the observable transitions in T :

Table 14.2 Mask of $\mathcal{W}$ (left), and description of the relevant observations and faults (right)

t	o	f
p_1	po	ε
p_2	po	ε
p_3	ε	$\boldsymbol{a}$
p_4	ε	$\boldsymbol{b}$
b_1	bo	ε
b_2	bo	ε
b_3	ε	$\boldsymbol{c}$
b_4	ε	$\boldsymbol{d}$
b_5	bo	ε
b_6	bo	ε
b_7	bo	$\boldsymbol{g}$
b_8	bo	$\boldsymbol{h}$

o	*Description*
po	The protection performs a normal action
bo	The breaker reacts (possibly abnormally) to a command

f	*Description*
$\boldsymbol{a}$	The protection issues the *cl* command instead of *op*
$\boldsymbol{b}$	The protection issues the *op* command instead of *cl*
$\boldsymbol{c}$	The breaker persists in staying closed upon the *op* command
$\boldsymbol{d}$	The breaker persists in staying open upon the *cl* command
$\boldsymbol{g}$	The breaker opens upon the *cl* command
$\boldsymbol{h}$	The breaker closes upon the *op* command

$$Obs(T) = [\, o \mid t \in T, (t, o, f) \in Msk(\mathcal{Y}), o \neq \varepsilon \,]. \tag{14.2}$$

T is said to *conform* with $\mathcal{O}$.

Example 4 According to $Msk(\mathcal{W})$ displayed in Table 14.2 and considering the trajectory $T = [p_3, b_5, p_1, b_3, p_4, b_3, p_2, b_5]$ of $\mathcal{W}$, we have $Obs(T) = [bo, po, po, bo]$.

A diagnosis is usually the set of faults involved in a trajectory; hence, it neither embodies any chronological order between faults nor records possible repetitions of the same fault. Still, defining a diagnosis as a set allows the domain of diagnoses to be finite, as it is bounded by the powerset of the domain of faults. The concept of a *temporal fault* introduced below contrasts with this view. Each trajectory can be associated with a temporal fault thanks to the mask.

Definition 3 The *temporal fault* of a trajectory T of a DES $\mathcal{Y}$ is the sequence of the faults associated with the faulty transitions in T:

$$Flt(T) = [\, f \mid t \in T, (t, o, f) \in Msk(\mathcal{Y}), f \neq \varepsilon \,]. \tag{14.3}$$

A *temporal-fault segment* is a contiguous subsequence of a temporal fault.

Example 5 Let $T = [p_3, b_5, p_1, b_3, p_4, b_3, p_2, b_5]$ be a trajectory of $\mathcal{W}$. Based on $Msk(\mathcal{W})$ shown in Table 14.2, the temporal fault relevant to T is $Flt(T) = [\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{b}, \boldsymbol{c}]$, where the faulty transitions occurred are p_3, b_3, p_4, and b_3.

Being, in general, the length of a trajectory unbounded, also a temporal observation and a temporal fault have usually an unbounded length. A diagnosis task is

given a temporal observation $\mathcal{O}$ of a DES $\mathcal{Y}$ and it has to find the corresponding temporal faults of $\mathcal{Y}$, which needs computing the trajectories of $\mathcal{Y}$ that conform with $\mathcal{O}$. Three different methods implement the diagnosis task: (a) with no knowledge compilation, namely *interpreted* diagnosis, (b) with total knowledge compilation, namely *compiled* diagnosis, and (c) with partial knowledge compilation, namely *hybrid* diagnosis. Technique (b) compiles offline the model of the DES into knowledge structures whose purpose is to boost the speed of the online diagnosis engine. Technique (c) produces offline an initial partial knowledge structure, which the online diagnosis engine may extend many times.

14.4 Diagnosis Problem with Temporal Faults

We assume that the relevant DES has performed a trajectory ending in a quiescent state. The trajectory of the DES manifests itself to an external observer as a temporal observation $\mathcal{O}$ (Definition 2). Since the actual trajectory that the DES has followed is unknown, the diagnosis task needs to infer the trajectories of the DES that conform with $\mathcal{O}$ and end in a quiescent state. The set of these trajectories is possibly infinite; hence, also the set of temporal faults corresponding to them is possibly infinite. This set is the diagnosis output, called the *candidate set* of $\mathcal{O}$.

Definition 4 Given a temporal observation $\mathcal{O}$ of a DES $\mathcal{Y}$, the relevant *candidate set* $\Delta(\mathcal{O})$ is the set of temporal faults associated with the trajectories that conform with $\mathcal{O}$:

$$\Delta(\mathcal{O}) = \{\, Flt(T) \mid T \in Space(\mathcal{Y}), Obs(T) = \mathcal{O} \,\}. \tag{14.4}$$

Example 6 A temporal observation $\mathcal{O} = [bo, po, po, bo]$ of $\mathcal{W}$ (cf. Example 4) is given. In order to produce $\Delta(\mathcal{O})$, the computation of the set $\mathbb{T}$ of trajectories T of $\mathcal{W}$ such that $Obs(T) = \mathcal{O}$ is needed (Eq. (14.4)). $\mathbb{T}$ is a regular expression on the component transitions (see $Space(\mathcal{W})$ in Fig. 14.2 and $Msk(\mathcal{W})$ in Table 14.2):

$$\mathbb{T} = p_3 b_5 p_1 b_3 (p_4 b_3)^* p_2 \left(b_5 \mid b_7 (p_3 b_4)^*\right)$$

whose language includes $T = [p_3, b_5, p_1, b_3, p_4, b_3, p_2, b_5]$ (cf. Example 4). Subsequently, $\Delta(\mathcal{O})$ can be inferred from $\mathbb{T}$ by substituting the faulty transitions with the relevant faults in $Msk(\mathcal{W})$, and then deleting the empty faults:

$$\Delta(\mathcal{O}) = \boldsymbol{a}\varepsilon\varepsilon\boldsymbol{c}\,(\boldsymbol{bc})^*\,\varepsilon\left(\varepsilon \mid \boldsymbol{g}\,(\boldsymbol{ad})^*\right) = \boldsymbol{ac}\,(\boldsymbol{bc})^*\left(\boldsymbol{g}\,(\boldsymbol{ad})^*\right)?$$

Similarly, with $\mathcal{O}' = [bo, po]$, we have $\mathbb{T}' = p_3 b_5 p_1 b_3 (p_4 b_3)^*$ and $\Delta(\mathcal{O}') = \boldsymbol{ac}\,(\boldsymbol{bc})^*$. Finally, with $\mathcal{O}'' = [po, bo]$, we have $\mathbb{T}'' = p_1\,(b_3 p_4)^*\, b_1$ and hence $\Delta(\mathcal{O}'') = (\boldsymbol{cb})^*$.

The ultimate goal of the a posteriori diagnosis of a DES is to generate the candidate set corresponding to a given temporal observation. However, the same candidate

set may be determined by means of different techniques, based on the amount of knowledge, compiled offline, that can be processed online by the diagnosis engine. On the one hand, the larger the compiled knowledge the diagnosis engine can exploit online, the more efficient the generation of the candidate set in terms of computational time. On the other, the size of the knowledge to be compiled may be so large that the process of knowledge compilation may become overwhelmingly impractical as far as memory allocation and computational time are concerned.

In the next sections we introduce three a posteriori diagnosis techniques for the computation of a candidate set. A first (*interpreted*) technique (Sect. 14.4.1) operates online only; it generates the candidate set by performing low-level reasoning on the DES structure and the models of the components, without the need of any compiled knowledge. A second (*compiled*) technique (Sect. 14.4.2) relies on *total* knowledge compilation: based on the DES specification, a (complete) *temporal diagnoser* is constructed offline, which allows for the fast online generation of a candidate set by the diagnosis engine. A third (*hybrid*) technique (Sect. 14.4.3) relies on *partial* knowledge compilation: only a partial temporal diagnoser is constructed offline, which is then exploited online. If and when it is needed, the partial temporal diagnoser is extended on the fly by the diagnosis engine, thereby possibly converging towards a (complete) temporal diagnoser over time.

14.4.1 Interpreted Technique

When no compiled knowledge is available, the engine for diagnosis of temporal faults is expected to compute the candidate set based solely on the DES description in terms of components and links. This means that no additional data structure has been generated offline to boost the computational efficiency online, including the space of the DES. This assumption may be regarded as an inconsistency if we look at Eq. (14.4), where the candidate set is defined in terms of trajectories of $\mathcal{Y}$, which in turn refer to the space of $\mathcal{Y}$. This problem may be overcome by considering that the actual trajectories involved in the computation of $\Delta(\mathcal{O})$ are those conforming with $\mathcal{O}$. Therefore, it suffices to materialize (online) the so-called $\mathcal{O}$ space of $\mathcal{Y}$, i.e. the part of $Space(\mathcal{Y})$ that conforms with $\mathcal{O}$, rather than the entire space.

Definition 5 The $\mathcal{O}$ space of a temporal observation $\mathcal{O}$ of $\mathcal{Y}$, $Osp(\mathcal{Y}, \mathcal{O})$, is a DFA whose regular language is the set of the trajectories T of $\mathcal{Y}$ where $Obs(T) = \mathcal{O}$.

$Osp(\mathcal{Y}, \mathcal{O})$ is computed by processing $\mathcal{O} = [o_1, \ldots, o_n]$ and the model of $\mathcal{Y}$. In $Osp(\mathcal{Y}, \mathcal{O})$, a state is a pair (x, i), x being a state of $\mathcal{Y}$ and i an *index* falling in $[0..n]$. The transition function is generated by accounting for the component transitions that can be triggered in a given state, starting from the initial one $(x_0, 0)$. Whenever a transition t can be triggered in a state (x, i) and reaches state x' of $\mathcal{Y}$, a transition $\langle (x, i), t, (x', (i + 1)) \rangle$ is generated in $Osp(\mathcal{Y}, \mathcal{O})$, provided that t is observable, $(t, o_{i+1}, f) \in Msk(\mathcal{Y})$ and $i < n$. If t is unobservable, however, the index i is not incremented. A state (x, n) is final iff x is quiescent.

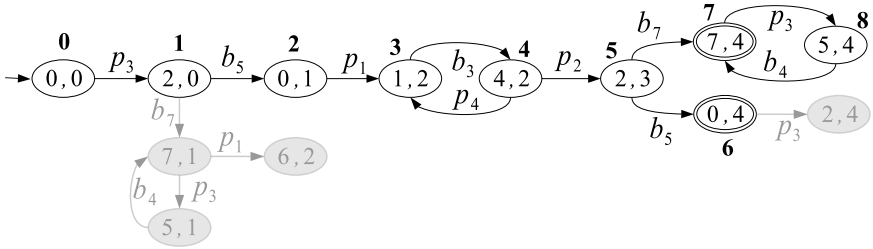

Fig. 14.3 $Osp(\mathcal{W}, \mathcal{O})$, the $\mathcal{O}$ space of the watcher $\mathcal{W}$, where $\mathcal{O} = [bo, po, po, bo]$

Example 7 $\mathcal{O} = [bo, po, po, bo]$ is the temporal observation of $\mathcal{W}$ already dealt with both in Example 4 and in Example 6. The generation of $Osp(\mathcal{W}, \mathcal{O})$ is traced in Fig. 14.3, where the gray portion is spurious (and eventually discarded), in that no path of transitions intersecting this part ends in a quiescent state. Therefore, the actual $Osp(\mathcal{W}, \mathcal{O})$ is that depicted in black. A state in $Osp(\mathcal{W}, \mathcal{O})$ is qualified as a pair (w, i), where w is a state in $Space(\mathcal{W})$ (Fig. 14.2),[2] while i is an index of $\mathcal{O}$, in the range [0..4]. The states of $Osp(\mathcal{W}, \mathcal{O})$ are renamed $0 \ldots 8$, with 0 being the initial state, while 6 and 7 are the final states, where $i = 4$ and p is a quiescent state of $\mathcal{W}$. It can be easily verified that the language[3] of $Osp(\mathcal{W}, \mathcal{O})$ is equal to the set of trajectories conforming with $\mathcal{O}$ in $Space(\mathcal{W})$.

According to Eq. (14.4) and Definition 5, a trajectory T is in $Osp(\mathcal{Y}, \mathcal{O})$ iff T is in $Space(\mathcal{Y})$ and $Obs(T) = \mathcal{O}$; hence, $Flt(T) \in \Delta(\mathcal{O})$. In fact, $\Delta(\mathcal{O})$ is the set of temporal faults inherent to all the trajectories in $Osp(\mathcal{Y}, \mathcal{O})$. However, considering all the trajectories in $Osp(\mathcal{Y}, \mathcal{O})$ to obtain $\Delta(\mathcal{O})$ is impractical since the number of such trajectories is possibly infinite. Still, a formal property of $\Delta(\mathcal{O})$ is essential to overcome this computational difficulty. In fact, the candidate set of a temporal observation $\mathcal{O}$ of a DES $\mathcal{Y}$, $\Delta(\mathcal{O})$, is a regular language over the domain of faults embodied in $Msk(\mathcal{Y})$.

Example 8 Consider the $\mathcal{O}$ space of $\mathcal{W}$ in Example 7 (Fig. 14.3), where the states are identified by $0 \ldots 8$. By substituting each component transition t marking an arc in $Osp(\mathcal{W}, \mathcal{O})$ with the (possibly empty) fault relevant to t in $Msk(\mathcal{W})$ (Table 14.2), we obtain the NFA in Fig. 14.4. Hence, a temporal fault is a string belonging to the language of this NFA. The infinite set of temporal faults identified below is the candidate set:

---

[2] The qualification of w by a number identifying a state of $Space(\mathcal{W})$ is only a shorthand, which does not assume, nor does it require, the materialization of $Space(\mathcal{W})$. In fact, w is represented as a triple $(\bar{p}, \bar{b}, e)$, $\bar{p}$ being a state of the protection, $\bar{b}$ being a state of the breaker, and e being a (possibly empty) event within the link.

[3] Intuitively, given an FA, its regular language is the set of strings over its alphabet that can be produced by following the transition function from the initial state to a final state. Hence, each string in the language of $Osp(\mathcal{W}, \mathcal{O})$ is a trajectory of $\mathcal{W}$ as each symbol in the alphabet of $Osp(\mathcal{W}, \mathcal{O})$ is a (component) transition.

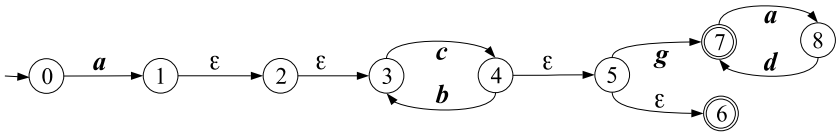

Fig. 14.4 NFA generated from $Osp(\mathcal{W}, \mathcal{O})$ in Fig. 14.3, by substituting the transitions with the relevant faults, as specified in $Msk(\mathcal{W})$ (Table 14.2)

$$\Delta(\mathcal{O}) = \boldsymbol{ac}(\boldsymbol{bc})^*(\boldsymbol{g}(\boldsymbol{ad})^*)?$$

The inspection of the NFA obtained from the $\mathcal{O}$ space by replacing the symbols marking the transitions has allowed us to specify the regular expression for the candidate set in Example 8. In general, however, a technique that can process this NFA to automatically generate the regular expression of $\Delta(\mathcal{O})$ is needed. Algorithm CANDIDATES, introduced in the next section, serves this purpose.

14.4.1.1 Algorithm CANDIDATES

To automatically generate the regular expression of $\Delta(\mathcal{O})$, an algorithm [9] proposed in the domain of the state diagrams of sequential circuits is adapted. Its input is an NFA and its output is the regular expression of the language this NFA accepts. This is quite helpful to implement the generation process for the regular language of $\Delta(\mathcal{O})$.

Algorithm 1 (lines 1–24) lists the pseudocode of the adapted algorithm, called CANDIDATES. Its input is an $\mathcal{O}$ space of a DES $\mathcal{Y}$ and its output is a regular expression $\mathcal{R}$ whose alphabet is the set of faults of $\mathcal{Y}$ and whose language is $\Delta(\mathcal{O})$. First, each component transition is replaced with the corresponding (possibly empty) fault defined in the mask $Msk(\mathcal{Y})$ (lines 1–2). Then, in lines 3–8, a new initial state α_0 and a new (single) final state α_q are possibly inserted along with an ε-transition from α_0 to the original initial state β_0 and an ε-transition from each original final state to α_q. This way, an NFA $\mathcal{N}$ having n_0 and n_q as initial and final state, respectively, is obtained (line 9). The algorithm has now to transform $\mathcal{N}$ into a new NFA that consists just in states n_0 and n_q and the transition $\langle n_0, r, n_q \rangle$, where r is a regular expression identifying $\Delta(\mathcal{O})$. The key idea in [9] is to progressively eliminate states and transitions in $\mathcal{N}$ while preserving the regular language that is accepted by the NFA. For this purpose, the alphabet of the NFA, which is initially the set of faults, is changed into a set of regular expressions on such faults. Three simplification rules are coded within the loop in lines 10–23. The first rule is embodied in lines 11–12: it replaces a sequence of transitions, with intermediate states that are neither entered nor exited by other transitions, with a single transition whose regular expression is the concatenation of the regular expressions marking the original transitions. The second rule (lines 13–14) substitutes some (parallel) transitions with a single transition whose regular expression is the alternative of the regular expressions relevant to the original transitions. The last rule (lines 16–23) removes both an internal node n of $\mathcal{N}$ and

Algorithm 1: CANDIDATES

input : $\mathcal{C}$, the $\mathcal{O}$ space of $\mathcal{Y}$, namely $Osp(\mathcal{Y}, \mathcal{O})$
output: $\mathcal{R}$, a regular expression identifying the language of the candidate set $\Delta(\mathcal{O})$

1 **foreach** *transition* $\langle \beta, t, \beta' \rangle$ *in* $\mathcal{C}$ **do**
2 Substitute f for t, where $(t, o, f) \in Msk(\mathcal{Y})$
3 **if** *there is a transition entering the initial state* β_0 *of* $\mathcal{C}$ **then**
4 Add both a new initial state α_0 and a new ε-transition $\langle \alpha_0, \varepsilon, \beta_0 \rangle$
5 **if** *there exist several final states or there is a transition exiting the (unique) final state* **then**
6 Add a new final state α_q
7 **foreach** *original final state* β_q *of* $\mathcal{C}$ **do**
8 Add a new ε-transition $\langle \beta_q, \varepsilon, \alpha_q \rangle$
9 Let $\mathcal{N}$ be the NFA so obtained from $\mathcal{C}$, with initial state n_0 and final state n_q
10 **while** $\mathcal{N}$ *includes more than one transition* **do**
11 **if** *there is a sequence* $Q = [\langle n, r_1, n_1 \rangle, \langle n_1, r_2, n_2 \rangle, \ldots, \langle n_{k-1}, r_k, n' \rangle]$ *of transitions,* $k \geq 2$*, where each state* $n_i, i \in [1..(k-1)]$*, has no further entering or exiting transitions* **then**
12 Replace Q with the transition $\langle n, (r_1 r_2 \cdots r_k), n' \rangle$
13 **else if** *there is a set* $S = \{\langle n, r_1, n' \rangle, \langle n, r_2, n' \rangle, \ldots, \langle n, r_k, n' \rangle\}$ *of transitions from n to* n' **then**
14 Replace S with the transition $\langle n, (r_1 | r_2 | \cdots | r_k), n' \rangle$
15 **else**
16 Let n be an internal (neither initial nor final) state of $\mathcal{N}$
17 **foreach** *transition* $\langle n', r', n \rangle$ *entering n, where* $n' \neq n$ **do**
18 **foreach** *transition* $\langle n, r'', n'' \rangle$ *exiting n, where* $n'' \neq n$ **do**
19 **if** *there is a loop transition* $\langle n, r, n \rangle$ *for n* **then**
20 Add a new transition $\langle n', (r'(r)^* r''), n'' \rangle$
21 **else**
22 Add a new transition $\langle n', (r' r''), n'' \rangle$
23 Remove n and all its entering / exiting transitions
24 $\mathcal{R} \leftarrow r$, where $\langle n_0, r, n_q \rangle$ is the unique transition remained in $\mathcal{N}$.

its entering / exiting transition, and adds a set of equivalent transitions. In case there exists a loop transition $\langle n, r, n \rangle$ for n, a new transition $\langle n', (r'(r^*)r''), n'' \rangle$ is created for each transition $\langle n', r', n \rangle$ entering n and for each transition $\langle n, r'', n'' \rangle$ exiting n. If there is no loop transition $\langle n, r, n \rangle$, then the new transition that is added is $\langle n', (r'r''), n'' \rangle$ instead. Eventually, n and all the transitions that enter or exit it are removed. The **while** loop terminates when there is just one transition $\langle n_0, r, n_q \rangle$, where r identifies in fact the regular language of $\Delta(\mathcal{O})$ (line 24).

Example 9 Consider the O space of $\mathcal{W}$ in Fig. 14.3. Traced in Fig. 14.5 is the transformation of $Osp(\mathcal{W}, \mathcal{O})$ carried out by the CANDIDATES algorithm in order to generate a regular expression identifying $\Delta(\mathcal{O})$. The first row of Fig. 14.5 accommodates the NFA $\mathcal{N}$ obtained by CANDIDATES in line 9, the newly-inserted final state being 9. Then, the loop in lines 10–23 is entered, where the three simplification rules are applied. Since the condition in line 11 holds, the sequence of transitions

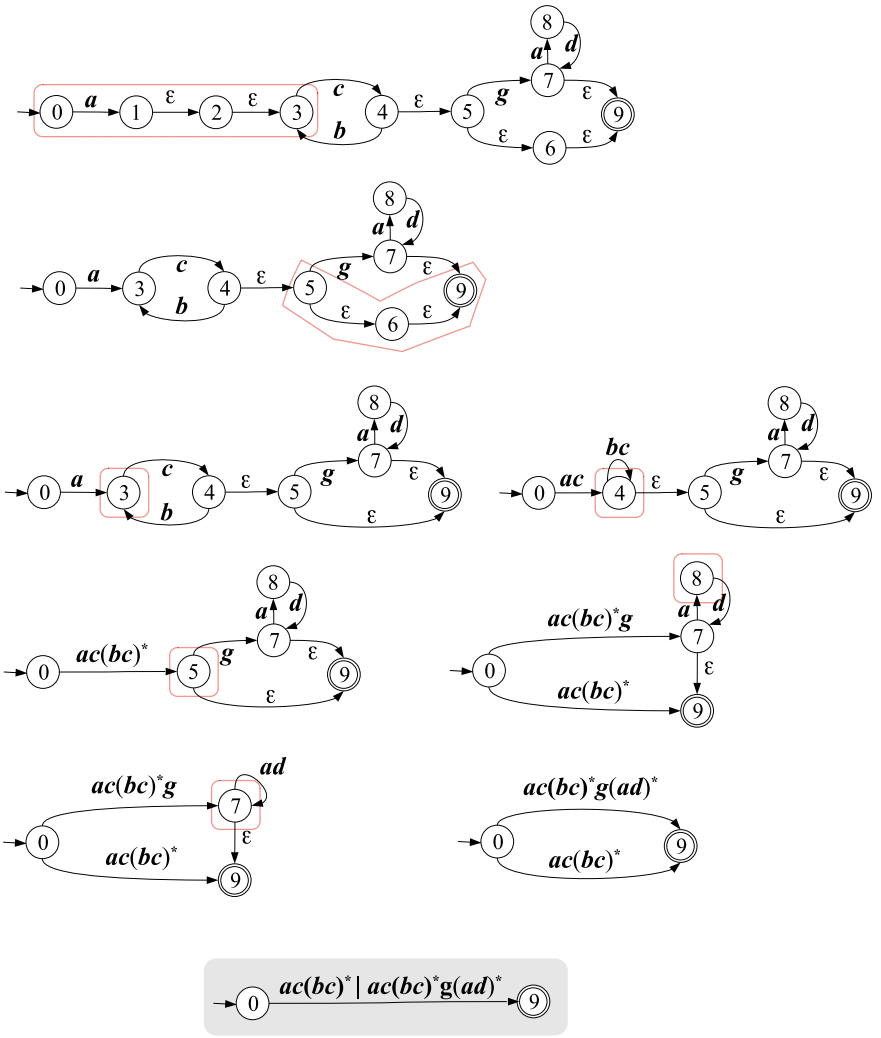

Fig. 14.5 Tracing of the CANDIDATES Algorithm applied to $Osp(\mathcal{W}, \mathcal{O})$ (cf. Figs. 14.3 and 14.4)

$[\langle 0, \boldsymbol{a}, 1\rangle, \langle 1, \varepsilon, 2\rangle, \langle 2, \varepsilon, 3\rangle]$ (highlighted in Fig. 14.5) is replaced by the single transition $\langle 0, \boldsymbol{a}, 3\rangle$, thereby leading to the NFA in the second row. Next, the sequence $[\langle 5, \varepsilon, 6\rangle, \langle 6, \varepsilon, 9\rangle]$ is replaced by $\langle 5, \varepsilon, 9\rangle$. Subsequently, the second simplification rule applies several times, which eliminates in succession states 3, 4, 5, 8, and 7, thereby leading to the NFA on the right of the fifth row. Eventually, the second rule comes into play (NFA on the bottom of the figure), by replacing the pair of parallel transitions from 0 to 9 with the single transition marked with the regular expression

$$r = \boldsymbol{ac}(\boldsymbol{bc})^* \mid \boldsymbol{ac}(\boldsymbol{bc})^*\boldsymbol{g}(\boldsymbol{ad})^* = \boldsymbol{ac}(\boldsymbol{bc})^*(\boldsymbol{g}(\boldsymbol{ad})^*)?$$

which is in fact the regular expression $\mathcal{R}$ identifying language $\Delta(\mathcal{O})$ (line 24). As expected, $\mathcal{R}$ is the same regular expression computed in Example 6 based on Definition 4.[4]

Proposition 1 *Algorithm* CANDIDATES *is sound and complete.*

Proof According to Definition 5, the language of $\mathcal{C} = Osp(\mathcal{Y}, \mathcal{O})$ consists in the set of trajectories T in $Space(\mathcal{Y})$ where $Obs(T) = \mathcal{O}$. After the substitutions performed in lines 1–2, the language of the NFA is the set of temporal faults $Flt(T)$ where $Obs(T) = \mathcal{O}$, in other words, it is $\Delta(\mathcal{O})$ (cf. Definition 4). When line 9 is reached, the language of $\mathcal{N}$ is still $\Delta(\mathcal{O})$, although a new initial state and a new final state have possibly been added. Also the reduction of $\mathcal{N}$ to a single transition $\langle n_0, r, n_q \rangle$ in lines 10–23 maintains the language of $\mathcal{N}$ within r, hence the latter identifies $\Delta(\mathcal{O})$. □

Two alternative methods that, differently from the interpreted technique, exploit knowledge compilation for diagnosis of temporal faults are presented in Sects. 14.4.2 and 14.4.3, respectively.

14.4.2 Compiled Technique

The interpreted technique for generating $\Delta(\mathcal{O})$ does not perform any (offline) preprocessing. Consequently, the algorithm CANDIDATES first generates (online) the $\mathcal{O}$ space, which is hardly desirable under tight time constraints. The compiled technique, instead, speeds up the diagnosis task by generating offline a *temporal diagnoser* (Definition 7) to be exploited online. Intuitively, the temporal diagnoser of a DES $\mathcal{Y}$, $Tdg(\mathcal{Y})$, is an NFA that a preprocessor draws from the specification of $\mathcal{Y}$ based on the mask of $\mathcal{Y}$. A set of triples $(o, \mathcal{L}, f)$ is the alphabet of $Tdg(\mathcal{Y})$: o is an observation of $\mathcal{Y}$, $\mathcal{L}$ is a regular language whose symbols are the faults of $\mathcal{Y}$, and f is a (possibly empty) fault. Each state of $Tdg(\mathcal{Y})$, called a *local space*, contains some local diagnosis information specified as regular expressions on faults. Remarkably, $Tdg(\mathcal{Y})$ enables one to generate $\Delta(\mathcal{O})$ efficiently.

Definition 6 Given a DES $\mathcal{Y}$ whose set of faults is $\mathbf{F}$, the *local space* of a state $\bar{x}$ of $\mathcal{Y}$, $Lsp(\bar{x})$, is an NFA:

$$Lsp(\bar{x}) = (\Sigma, X, \tau, x_0, X_e, X_q) \tag{14.5}$$

where $\Sigma = \mathbf{F} \cup \{\varepsilon\}$ is the alphabet, X is the set of the states in $Space(\mathcal{Y})$ that can be reached from $\bar{x}$ via (possibly empty) sequences of unobservable transitions, $x_0 = \bar{x}$

[4] What matters here is not the exact equality of the two regular expressions, however. Generally speaking, it is their *equivalence* that counts, inasmuch they denote the same language and, therefore, the same set of temporal faults.

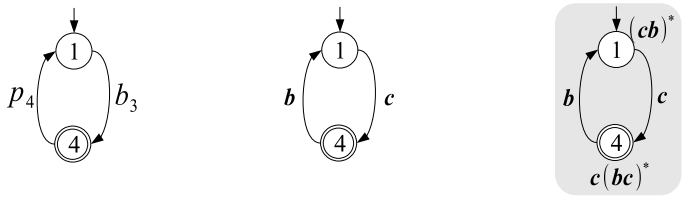

Fig. 14.6 Construction of $Lsp(1)$ in conformance with Definition 6, where 1 is a state of $Space(\mathcal{W})$ (cf. Fig. 14.2)

is the initial state, $X_e \subseteq X$ is the set of the exit states, where $x_e \in X_e$ iff $\langle x_e, t, x' \rangle$ is a transition in $Space(\mathcal{Y})$ and t is observable, $X_q \subseteq X$ is the set of states in X that are quiescent in $Space(\mathcal{Y})$, and $\tau : X \times \Sigma \mapsto 2^X$ is the transition function, where $\langle x_1, f, x_2 \rangle$ is an arc in τ iff $\langle x_1, t, x_2 \rangle$ is a transition in $Space(\mathcal{Y})$ and $(t, \varepsilon, f) \in Msk(\mathcal{Y})$.

Every state $x \in X_e \cup X_q$ is a *labeled state*, being marked with a regular expression whose language is the set of temporal-fault segments relevant to the trajectory segments in $Space(\mathcal{Y})$ from $\bar{x}$ to x, denoted $\mathcal{L}(x)$. The *diagnosis language* of $Lsp(\bar{x})$, namely $\mathcal{L}(Lsp(\bar{x}))$, is a regular language over $\mathbf{F}$, the set of faults of $\mathcal{Y}$:

$$\mathcal{L}(Lsp(\bar{x})) = \begin{cases} \emptyset & \text{if } X_q = \emptyset \\ \mathcal{L}(x) & \text{if } X_q = \{x\} \\ \mathcal{L}(x_1) \mid \ldots \mid \mathcal{L}(x_n) & \text{if } X_q = \{x_1, \ldots, x_n\}, n \geq 2. \end{cases} \tag{14.6}$$

In other words, a local space rooted in a state $\bar{x}$ of a DES $\mathcal{Y}$ comprehends the subgraph of $Space(\mathcal{Y})$ that contains transitions that are all unobservable, where each state x in $Space(\mathcal{Y})$ that either has an exiting observable transition or is quiescent is decorated with a regular expression whose language is the set of temporal-fault segments relevant to the sequences of transitions traversing the local space from $\bar{x}$ to x.

Example 10 Figure 14.6, on the left, shows the unobservable subspace of $Space(\mathcal{W})$ whose root is state $1 = (shorted, closed, op)$ (cf. Fig. 14.2). As described in the center of the figure, component transitions p_4 and b_3 are substituted with the faults associated with them in $Msk(\mathcal{W})$ (Table 14.2). Since 1 is an exit state and 4 is a quiescent state, both of them are marked with a regular expression involving faults $\boldsymbol{c}$ and $\boldsymbol{b}$, representing the temporal-fault segments from 1 to either 1 or 4, thus generating the local space $Lsp(1)$ on the right of Fig. 14.6. Eventually, the local space defines the diagnosis language $\mathcal{L}(Lsp(1)) = \mathcal{L}(4) = \boldsymbol{c}(\boldsymbol{bc})^*$.

The internal states of a local space cannot be marked with regular expressions by applying the algorithm CANDIDATES as the marking process addresses several states (namely, the labeled states $X_e \cup X_q$), each of which has, in general, its own different regular expression. However, once the corresponding faults have replaced

Algorithm 2: LOCAL SPACE

input : $\bar{x}$, a state in $Space(\mathcal{Y})$
output: $Lsp(\bar{x})$, the local space of $\bar{x}$

1 Set $Lsp(\bar{x})$ to the unobservable subspace of $Space(\mathcal{Y})$ whose root is $\bar{x}$, with $\bar{X} = \{\bar{x}_1, \ldots, \bar{x}_n\}$ being the set of states where each $\bar{x}_i$, $i \in [1..n]$, either has an exiting observable transition or is quiescent in $Space(\mathcal{Y})$
2 Substitute each transition $\langle x, t, x' \rangle$ in $Lsp(\bar{x})$ with $\langle x, f, x' \rangle$, where $(t, \varepsilon, f) \in Msk(\mathcal{X})$
3 $\mathcal{N} \leftarrow Lsp(\bar{x})$
4 Add in $\mathcal{N}$ a new initial state α_0 and an ε-transition $\langle \alpha_0, \varepsilon, \bar{x} \rangle$
5 **foreach** *state* $\bar{x}_i \in \bar{X}$ *of* $\mathcal{N}$, $i \in [1..n]$ **do**
6 Add a final state α_{q_i} and an ε-transition $\langle \bar{x}_i, \varepsilon, \alpha_{q_i} \rangle$
7 **while** *there is a state in* $\mathcal{N}$ *distinct from both the initial state* α_0 *any final state* α_{q_i}, $i \in [1..n]$, **or** *there are several transitions from* α_0 *to the same* α_{q_i}, $i \in [1..n]$ **do**
8 **if** *there is a sequence* $Q = [\langle x, r_1, x_1 \rangle, \langle x_1, r_2, x_2 \rangle, \ldots, \langle x_{k-1}, r_k, x' \rangle]$ *of transitions,* $k \geq 2$, *where each* x_i, $i \in [1..(k-1)]$, *has no further entering or exiting transitions* **then**
9 Replace Q with the transition $\langle x, (r_1 r_2 \cdots r_k), x' \rangle$
10 **else if** *there is a set* $S = \{\langle x, r_1, x' \rangle, \langle x, r_2, x' \rangle, \ldots, \langle x, r_k, x' \rangle\}$ *of transitions from* x *to* x', $k \geq 2$ **then**
11 Substitute the transition $\langle x, (r_1 | \cdots | r_k), x' \rangle$ for S
12 **else**
13 Let x be a state of $\mathcal{N}$ where $x \neq \alpha_0$ and $x \neq \alpha_{q_i}$, $i \in [1..n]$
14 **foreach** *transition* $\langle x', r', x \rangle$ *entering* x, *where* $x' \neq x$ **do**
15 **foreach** *transition* $\langle x, r'', x'' \rangle$ *exiting* x, *where* $x'' \neq x$ **do**
16 **if** *there is a loop transition* $\langle x, r, x \rangle$ *for* x **then**
17 Add a transition $\langle x', (r'(r)^* r''), x'' \rangle$ in $\mathcal{N}$
18 **else**
19 Add a transition $\langle x', (r' r''), x'' \rangle$ in $\mathcal{N}$
20 Remove x and all its entering/exiting transitions
21 **foreach** *transition* $\langle \alpha_0, r_i, \alpha_{q_i} \rangle$ *in* $\mathcal{N}$, $i \in [1..n]$ **do**
22 Mark the state $\bar{x}_i \in \bar{X}$ of $Lsp(\bar{x})$ with the regular expression r_i.

the component transitions, the algorithm LOCAL SPACE substantially performs on the NFA $\mathcal{N}$ the same actions as CANDIDATES.

14.4.2.1 Algorithm LOCAL SPACE

The input of LOCAL SPACE (lines 1–22) is a state $\bar{x}$ in $Space(\mathcal{Y})$ while its output is the local space $Lsp(\bar{x})$. This local space is initially the unobservable subspace of $Space(\mathcal{Y})$ whose root is $\bar{x}$, $\bar{X}$ being the set of labeled states of $Lsp(\bar{x})$. Once the corresponding (possibly empty) faults have replaced the component transitions (line 2), the NFA $\mathcal{N}$ is assigned a copy of the current instance of $Lsp(\bar{x})$. Subsequently, a new initial state α_0 and an ε-transition $\langle \alpha_0, \varepsilon, \bar{x} \rangle$ are inserted into $\mathcal{N}$ (line 4). Also, for each labeled state $\bar{x}_i, i \in [1..n]$, a new final state α_{q_i} and an ε-transition $\langle \bar{x}_i, \varepsilon, \alpha_{q_i} \rangle$

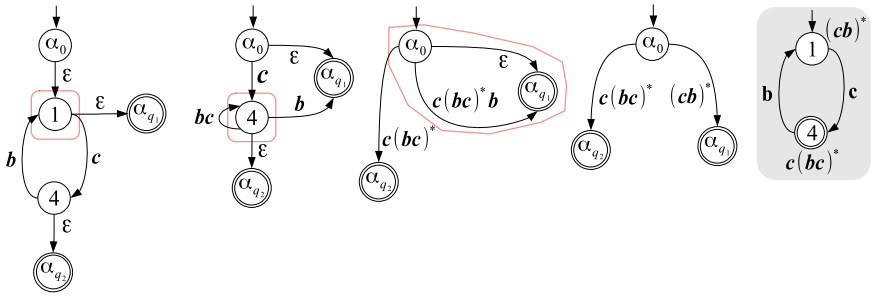

Fig. 14.7 Generation of *Lsp*(1) (cf. Fig. 14.6) by the LOCAL SPACE algorithm

are created in $\mathcal{N}$ (lines 5–6). Lines 7–20, which resemble almost entirely lines 10–23 of the algorithm CANDIDATES, perform the transformation of $\mathcal{N}$; while there exists a state in $\mathcal{N}$ that is neither initial nor final, the processing goes on (line 7). Once the loop has ended, all transitions in $\mathcal{N}$ exit the initial state and enter a final state, namely $\langle \alpha_0, r_i, \alpha_{q_i} \rangle, i \in [1..n]$, where r_i is the regular expression that marks the labeled state $\bar{x}_i$ in $Lsp(\bar{x})$.

Example 11 Figure 14.7 traces the generation of $Lsp(1)$ (cf. Fig. 14.6). The left of the figure displays the NFA $\mathcal{N}$ generated in line 3. The center of the figure shows the NFA after the (two step) removal of states 1 and 4. Now, the pair of transitions from α_0 to α_{q_1} are merged into one marked with $(\boldsymbol{cb})^*$, this being equivalent to $\boldsymbol{c}(\boldsymbol{bc})^*\boldsymbol{b} \mid \varepsilon$. Once the main loop has come to an end, lines 21–22 mark the states 1 and 4 of $Lsp(1)$ with the regular expressions relevant to the transitions in the final configuration of $\mathcal{N}$, that is, $(\boldsymbol{cb})^*$ and $\boldsymbol{c}(\boldsymbol{bc})^*$, respectively. Notice how the final automaton, depicted in Fig. 14.7 (right), equals the local space $Lsp(1)$ in Fig. 14.6.

A local space is a node of a *temporal diagnoser*, a structure containing knowledge that supports the efficient generation of candidate sets.

Definition 7 Let $\mathcal{Y}$ be a DES, $Space(\mathcal{Y}) = (\Sigma, X, \tau, x_0, X_q)$ its space, $\mathbf{O}$ the domain of its observations, and $\mathbf{F}$ the domain of its faults. Let $\mathbf{L}$ be the set of regular languages whose alphabet of symbols is $\mathbf{F}$. The *temporal diagnoser* of $\mathcal{Y}$, namely $Tdg(\mathcal{Y})$, is an NFA:

$$Tdg(\mathcal{Y}) = (\Sigma', X', \tau', x'_0, X'_q) \tag{14.7}$$

where $\Sigma' \subseteq \mathbf{O} \times \mathbf{L} \times (\mathbf{F} \cup \{\varepsilon\})$ is the alphabet, X' is the set of states, each state being the local space of a particular state in $Space(\mathcal{Y})$, $x'_0 = Lsp(x_0)$ is the initial state, X'_q is the set of quiescent states, namely the local spaces containing some quiescent state(s) in $Space(\mathcal{Y})$, and τ' is the transition function, $\tau' : (X' \times X) \times \Sigma' \mapsto 2^{(X' \times X)}$, where $\langle (x'_1, x_1), (o, \mathcal{L}(x_1), f), (x'_2, x_2) \rangle$ is an edge in τ' iff x_1 is a state in the local space x'_1, $\langle x_1, t, x_2 \rangle \in \tau$, $(t, o, f) \in Msk(\mathcal{Y})$ with $o \neq \varepsilon$, and $x'_2 = Lsp(x_2)$.

Intuitively, $Tdg(\mathcal{Y})$ is defined starting from the initial state $x'_0 = Lsp(x_0)$, where x_0 is the initial state of $\mathcal{Y}$. Given a state x'_1 of $Tdg(\mathcal{Y})$, there exists an arc from each

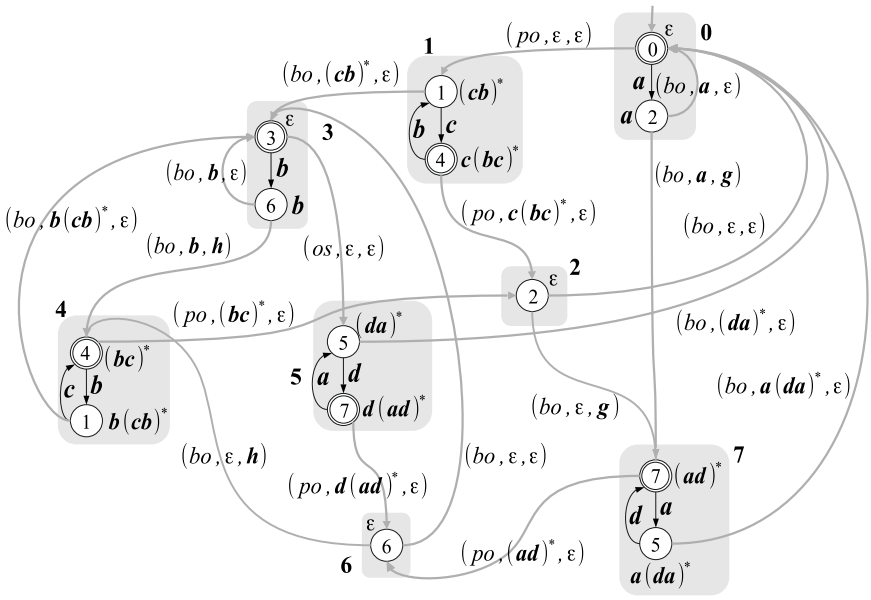

Fig. 14.8 *Tdg*($\mathcal{W}$), the temporal diagnoser of DES $\mathcal{W}$

exiting state x_1 in x_1' to the initial state x_2 of another (not necessarily different) local space $x_2' = Lsp(x_2)$ iff there exists an observable transition $\langle x_1, t, x_2 \rangle$ in $Space(\mathcal{Y})$. This arc is marked with a triple $(o, \mathcal{L}(x_1), f)$, where $(t, o, f) \in Msk(\mathcal{Y})$ and $\mathcal{L}(x_1)$ is the regular expression on temporal faults marking the state x_1 within x_1'. Notice how $Tdg(\mathcal{Y})$ is in general an NFA, since several arcs may exit the same local space, which are marked with the same triple and, specifically, the same observation.

Example 12 Shown in Fig. 14.8 is $Tdg(\mathcal{W})$ (cf. $Space(\mathcal{W})$ in Fig. 14.2), which contains eight states, namely $\mathbf{0} \ldots \mathbf{7}$. By chance, there is one state $Lsp(p)$ for each state p in $Space(\mathcal{W})$. The initial state $\mathbf{0} = Lsp(0)$ is exited by three edges: $\langle (\mathbf{0}, 0), (po, \varepsilon, \varepsilon), (\mathbf{1}, 1) \rangle$, $\langle (\mathbf{0}, 2), (bo, \boldsymbol{a}, \varepsilon), (\mathbf{0}, 0) \rangle$, and $\langle (\mathbf{0}, 2), (bo, \boldsymbol{a}, \boldsymbol{g}), (\mathbf{7}, 7) \rangle$. To avoid confusion, transitions between states of $Tdg(\mathcal{W})$ are represented as thick gray arrows, while transitions between states in local spaces appear as thin black arrows.

Once the temporal diagnoser of $\mathcal{Y}$ has been built offline, it is exploitable online by the algorithm FAST CANDIDATES to generate the candidate sets of temporal observations of $\mathcal{Y}$. FAST CANDIDATES is more efficient than CANDIDATES (cf. Sect. 14.4.1.1), since it avoids low-level model-based reasoning and it can directly access diagnosis information within local spaces and arcs.

14.4.2.2 Algorithm FAST CANDIDATES

The *total knowledge* assumption made by the FAST CANDIDATES algorithm is that the temporal diagnoser is available in its entirety. This, however, is realistic only for DESs having few components, as the number of states is exponentially increasing. Intuitively, to generate the candidate set $\Delta(\mathcal{O})$, the regular expressions relevant to the transitions in the temporal diagnoser $Tdg(\mathcal{Y})$ are concatenated based on the order of the observations in $\mathcal{O}$. Once the transition corresponding to the last observation in $\mathcal{O}$ has been traversed and a quiescent state (local space) x_q has been reached, the obtained regular expression is eventually appended to the (offline precomputed) diagnosis language $\mathcal{L}(x_q)$. Several paths in $Tdg(\mathcal{Y})$ can produce the same temporal observation $\mathcal{O}$, as $Tdg(\mathcal{Y})$ is an NFA; hence, in general the resulting regular expression consists in the alternative of several subexpressions.

The inputs of the FAST CANDIDATES algorithm (lines 1–16) are a temporal diagnoser $Tdg(\mathcal{Y})$ and a temporal observation $\mathcal{O}$ of $\mathcal{Y}$; its output is a regular expression $\mathcal{R}$ defining language $\Delta(\mathcal{O})$. A set of *contexts*, denoted $\mathbb{C}$, is used, where each context is a pair (x, r), x being a state (local space) of $Tdg(\mathcal{Y})$ and r a regular expression whose symbols are faults of $\mathcal{Y}$. In line 1, $\mathbb{C}$ contains just the context (x_0, ε) inherent to x_0, the initial state of $Tdg(\mathcal{Y})$. The ensuing loop (lines 2–11) scans, one by one, the observations in $\mathcal{O}$ in the given order; the current instance of $\mathbb{C}$ is processed to produce a new set of contexts $\mathbb{C}_{new}$ at each iteration. More precisely, for the current observation o, the nested loop (lines 4–10) takes into account each context (x', r') in

Algorithm 3: FAST CANDIDATES

input : $Tdg(\mathcal{Y}) = (\Sigma, X, \tau, x_0, X_q)$, the temporal diagnoser of a DES $\mathcal{Y}$
$\mathcal{O}$, a temporal observation of $\mathcal{Y}$
output: $\mathcal{R}$, a regular expression defining language $\Delta(\mathcal{O})$

```
1  ℂ ← {(x0, ε)}
2  foreach observation o ∈ 𝒪 do
3  |  ℂnew ← ∅
4  |  foreach (x', r') ∈ ℂ do
5  |  |  foreach arc ⟨(x', x), (o, r, f), (x'2, x2)⟩ in τ do
6  |  |  |  r2 ← r'rf
7  |  |  |  if (x'2, r'2) ∈ ℂnew then
8  |  |  |  |  Substitute (x'2, (r'2|r2)) for (x'2, r'2) in ℂnew
9  |  |  |  else
10 |  |  |  |  Insert (x'2, r2) into ℂnew
11 |  ℂ ← ℂnew
12 Remove from ℂ every context (x, r) where x ∉ Xq
13 if ℂ = {(x, r)} then
14 |  ℛ ← rℒ(x), where ℒ(x) is the diagnosis language of the local space x
15 else if ℂ = {(x1, r1), ..., (xk, rk)}, where k > 1 then
16 |  ℛ ← (r1 (ℒ (x1))) | ... | (rk (ℒ (xk))).
```

$\mathbb{C}$, and its inner loop (lines 5–10) browses each arc of $Tdg(\mathcal{Y})$ that exits from x' and is marked with a triple (o, r, f). In the body of the innermost loop (line 6), for the considered observation, context and arch, a regular expression $r_2 = r'rf$ is drawn, where r' encompasses the faults up to x', r encompasses the faults up to the state x inside x', and, as explained above, f is the (possibly empty) fault associated with the observable transition (generating o and) exiting x'. $\mathbb{C}_{\text{new}}$ is updated in lines 7–10, in a way that depends on whether $\mathbb{C}_{\text{new}}$ includes a context relevant to the reached state x_2' or not. If $\mathbb{C}_{\text{new}}$ includes a context (x_2', r_2'), then the regular expression r_2' of the latter is extended with the alternative r_2, thus obtaining the updated context $(x_2', (r_2'|r_2))$ (line 8). Otherwise, the context (x_2', r_2) is created and added to $\mathbb{C}_{\text{new}}$ (line 10). Before ending the iteration of the outermost loop, $\mathbb{C}_{\text{new}}$ is assigned to $\mathbb{C}$ (line 11). Once the execution of the outermost loop has ended, every context (x, r) such that $x \notin X_{\text{q}}$ is removed from $\mathbb{C}$ (line 12), as there does not exist any trajectory that reaches a quiescent state of x. Lines 13–16 determine the regular expression $\mathcal{R}$. If $\mathbb{C}$ contains only one context (x, r) (lines 13–14), $\mathcal{R}$ becomes the concatenation of r and the diagnosis language of x, that is, $\mathcal{R} = r\mathcal{L}(x)$. The reason for it is that r encompasses the faults up to the initial state of x, while $\mathcal{L}(x)$ encompasses the faults up to any quiescent state of $\mathcal{Y}$ within x. If, instead, $\mathbb{C}$ contains several contexts (lines 15–16), the above operation is performed for each context (x_i, r_i), $i \in [1..k]$; thus, the yielded regular expression $\mathcal{R}$ consists in all the alternatives $r_i(\mathcal{L}(x_i))$, $i \in [1..k]$.

Example 13 Considering Example 9, where the candidate set relevant to the temporal observation $\mathcal{O} = [bo, po, po, bo]$ of $\mathcal{W}$ is generated by the CANDIDATES algorithm, we now compute $\Delta(\mathcal{O})$ by means of FAST CANDIDATES. Traced in Table 14.3 is the computation of the set of contexts, namely $\mathbb{C}$, for each observation $o_i \in \mathcal{O}$, $i \in [0..4]$. Starting from the singleton $\{(\mathbf{0}, \varepsilon)\}$ (first row in Table 14.3), the next instance of $\mathbb{C}$, which corresponds to the first observation bo, is determined by focusing on the transitions of $Tdg(\mathcal{W})$ (cf. Fig. 14.8) that are marked with a triple involving the observation bo, namely $\langle(\mathbf{0}, 2), (bo, \boldsymbol{a}, \varepsilon), (\mathbf{0}, 0)\rangle$ and $\langle(\mathbf{0}, 2), (bo, \boldsymbol{a}, \boldsymbol{g}), (\mathbf{7}, 7)\rangle$. According to lines 6–10 of FAST CANDIDATES, the new set of contexts is $\mathbb{C}_{\text{new}} = \{(\mathbf{0}, \boldsymbol{a}), (\mathbf{7}, \boldsymbol{ag})\}$ (second row in Table 14.3). Eventually, after the last observation bo (last row in Table 14.3), since both states $\mathbf{0}$ and $\mathbf{7}$ are in X_{q}, no context is removed from $\mathbb{C} = \{(\mathbf{0}, \boldsymbol{ac}(\boldsymbol{bc})^*), (\mathbf{7}, \boldsymbol{ac}(\boldsymbol{bc})^*\boldsymbol{g})\}$. Hence, based on line 16:

$$\mathcal{R} = \boldsymbol{ac}(\boldsymbol{bc})^* \underbrace{\varepsilon}_{\mathcal{L}(\mathbf{0})} \mid \boldsymbol{ac}(\boldsymbol{bc})^*\boldsymbol{g} \underbrace{(\boldsymbol{ad})^*}_{\mathcal{L}(\mathbf{7})} = \boldsymbol{ac}(\boldsymbol{bc})^*(\boldsymbol{g}(\boldsymbol{ad})^*)?$$

Notably, $\mathcal{R}$ equals the regular expression generated by CANDIDATES in Example 9. More generally, the correctness of FAST CANDIDATES is proven in Proposition 2.

Proposition 2 *Algorithm* FAST CANDIDATES *is sound and complete.*

Proof The completeness of FAST CANDIDATES is proven if we can show that this algorithm computes a language $\mathcal{R}$ that equals the candidate set $\Delta(\mathcal{O})$ defined in Eq. (14.4), where $\mathcal{O} = [o_1, \ldots, o_n]$. Notice that, by unfolding the local spaces in

Table 14.3 Tracing of the FAST CANDIDATES algorithm based on $\mathcal{O} = [bo, po, po, bo]$

i	o_i	$\mathcal{O}_i$	$\mathbb{C}$
0		[ ]	$\{(\mathbf{0}, \varepsilon)\}$
1	bo	$[bo]$	$\{(\mathbf{0}, \boldsymbol{a}), (\mathbf{7}, \boldsymbol{ag})\}$
2	po	$[bo, po]$	$\{(\mathbf{1}, \boldsymbol{a}), (\mathbf{6}, \boldsymbol{ag}(\boldsymbol{ad})^*)\}$
3	po	$[bo, po, po]$	$\{(\mathbf{2}, \boldsymbol{ac}(\boldsymbol{bc})^*)\}$
4	bo	$[bo, po, po, bo]$	$\{(\mathbf{0}, \boldsymbol{ac}(\boldsymbol{bc})^*), (\mathbf{7}, \boldsymbol{ac}(\boldsymbol{bc})^*\boldsymbol{g})\}$

the temporal diagnoser $Tdg(\mathcal{Y})$, we obtain a graph that resembles $Space(\mathcal{Y})$, where each identifier t of a component transition in $Space(\mathcal{Y})$ is replaced with either a (possibly empty) fault, within a state of $Tdg(\mathcal{Y})$, or with a triple (o, r, f), between states of $Tdg(\mathcal{Y})$.[5] Each component transition in a state (local space) of $Tdg(\mathcal{Y})$ is unobservable. The regular expression r labeling each state x within a local space x' denotes all the strings of faults inherent to the trajectory segments of $\mathcal{Y}$ from the initial state of x' to x. According to Eq. (14.4), if $\mathcal{F}$ is a temporal fault in $\Delta(\mathcal{O})$, then a trajectory T in $Space(\mathcal{Y})$ is such that $\mathcal{F} = Flt(T)$ and $Obs(T) = \mathcal{O}$. In $Tdg(\mathcal{Y})$, the trajectory T is traced by a path $\wp$ from the initial state of the local space representing the initial state of $Tdg(\mathcal{Y})$ to a quiescent state x_q of a (quiescent) state x'_q of $Tdg(\mathcal{Y})$. A component transition in T that produces an observation o in $\mathcal{O}$ corresponds to a transition marked with (o, r, f) in $Tdg(\mathcal{Y})$. FAST CANDIDATES outputs a regular expression $\mathcal{R}$ that certainly accounts for the path $\wp$, as it is the observations in $\mathcal{O}$ that drive the construction of $\mathbb{C}$. In other words, there is an alternative in $\mathcal{R}$ that is a regular expression $r_\wp$ constructed based on $\wp$ by concatenating the regular expressions r_i associated with the observations o_i, $i \in [1..n]$, within the triples (o_i, r_i, f_i) marking the transitions of $Tdg(\mathcal{Y})$, and appending this concatenation with $\mathcal{L}(x'_q)$, the diagnosis language of the quiescent state x'_q of $Tdg(\mathcal{Y})$. The temporal fault $\mathcal{F}$ is necessarily included in the language of $r_\wp$ since each r_i encompasses the (segments of) temporal faults from a state to the next one. This concludes the proof of completeness. Soundness is proven by showing that assuming that $\mathcal{F}$ is a string in the language of $\mathcal{R}$ implies that $\mathcal{F}$ is a temporal fault in $\Delta(\mathcal{O})$. $\mathcal{F}$ is generated by following of a path $\wp$ that traverses the unfolded $Tdg(\mathcal{Y})$, from the initial state of the initial state of $Tdg(\mathcal{Y})$ to a quiescent state of a quiescent state of $Tdg(\mathcal{Y})$, fulfilling the constraint that the subsequence of (external) transitions of $Tdg(\mathcal{Y})$ in $\wp$ produces the sequence of observations in $\mathcal{O}$. Path $\wp$ corresponds to a trajectory T in $Space(\mathcal{Y})$, where $Obs(T) = \mathcal{O}$ and $Flt(T) = \mathcal{F}$; hence, based on Eq. (14.4), $\mathcal{F} \in \Delta(\mathcal{O})$. □

As already remarked, this section has dealt with a diagnosis technique that is dubbed *compiled* as the *whole* temporal diagnoser is assumed to be available. However, this assumption is not realistic in most cases owing to the complexity of the construction of the temporal diagnoser. Hence, the next section will present a method,

[5] This does not mean that the unfolding of $Tdg(\mathcal{Y})$ is isomorphic to $Space(\mathcal{Y})$, as a state in $Space(\mathcal{Y})$ may appear several times in different states (local spaces) of $Tdg(\mathcal{Y})$. Rather, what is preserved are the trajectories of $\mathcal{Y}$.

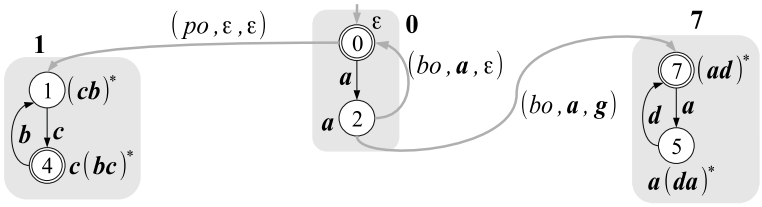

Fig. 14.9 *Pdg*($\mathcal{W}$), a partial temporal diagnoser of $\mathcal{W}$ (cf. Fig. 14.8)

dubbed *hybrid*, that does not rely on a complete knowledge compilation, instead it adopts only a partial one. An (even small) portion of the temporal diagnoser is generated offline and then extended online, if an when needed to compute a specific candidate set.

14.4.3 Hybrid Technique

The generation of a complete temporal diagnoser is impractical for real size DESs, owing to the exponential growth of the number of its states. Therefore, the proposed hybrid technique generates a *partial temporal diagnoser* in advance; this structure may later be expanded either offline, based on meaningful *behavioral scenarios*, as illustrated in [25], or online. The construction of *Space*($\mathcal{Y}$) is infeasible for the same reasons. Therefore, hereafter, when dealing with *Space*($\mathcal{Y}$), a notation like $\langle x, t, x' \rangle$ does not involve that *Space*($\mathcal{Y}$) is materialized: this is just a formal assertion that transition t can be triggered when the state of $\mathcal{Y}$ is x, thus moving $\mathcal{Y}$ to state x'.

Definition 8 A *partial temporal diagnoser* of a DES $\mathcal{Y}$, denoted *Pdg*($\mathcal{Y}$), is a connected subgraph of the temporal diagnoser *Tdg*($\mathcal{Y}$) that contains the initial state of *Tdg*($\mathcal{Y}$).

Example 14 Given the DES $\mathcal{W}$ that we have considered as a case study, Fig. 14.8 displays its temporal diagnoser while Fig. 14.9 shows a partial temporal diagnoser, which contains three states, i.e. **0**, **1**, and **7**, and three transitions, i.e. $\langle(\mathbf{0}, 0), (po, \varepsilon, \varepsilon), (\mathbf{1}, 1)\rangle$, $\langle(\mathbf{0}, 2), (bo, \boldsymbol{a}, \varepsilon), (\mathbf{0}, 0)\rangle$, and $\langle(\mathbf{0}, 2), (bo, \boldsymbol{a}, \boldsymbol{g}), (\mathbf{7}, 7)\rangle$.

Algorithm FAST CANDIDATES specified in Sect. 14.4.2.2 has been revisited in order to cope with a partial temporal diagnoser. Specifically, each state x' in *Pdg*($\mathcal{Y}$) needs to have a complete transition function as far as the observation o is concerned before running the loop in line 5 of FAST CANDIDATES.

14.4.3.1 Algorithm HYBRID CANDIDATES

The algorithm HYBRID CANDIDATES (lines 1–21) is different from FAST CANDIDATES mainly for a couple of reasons. First, its input parameter is a *partial* temporal

Algorithm 4: HYBRID CANDIDATES

input : $Pdg(\mathcal{Y}) = \left(\Sigma, X, \tau, x_0^p, X_q\right)$, a partial temporal diagnoser of a DES $\mathcal{Y}$
$\mathcal{O}$, a temporal observation of $\mathcal{Y}$
output: $\mathcal{R}$, a regular expression defining the candidate set $\Delta(\mathcal{O})$
As a side effect, $Pdg(\mathcal{Y})$ is upgraded based on $\mathcal{O}$

1 $\mathbb{C} \leftarrow \{(x_0^p, \varepsilon)\}$
2 **foreach** *observation* $o \in \mathcal{O}$ **do**
3 $\quad \mathbb{C}_{\text{new}} \leftarrow \emptyset$
4 $\quad$ **foreach** $(x^p, r^p) \in \mathbb{C}$ **do**
5 $\quad\quad$ **if** *there is no arc* $\langle (x^p, x), (o, r, f), (x_2^p, x_2)\rangle$ *in* τ **then**
6 $\quad\quad\quad$ **foreach** $x \in x^p$, $\langle x, t, x_2\rangle$ *is a transition in* $Space(\mathcal{Y})$, $(t, o, f) \in Msk(\mathcal{Y})$ **do**
7 $\quad\quad\quad\quad$ **if** $Lsp(x_2)$ *is not a state in* $Pdg(\mathcal{Y})$ **then**
8 $\quad\quad\quad\quad\quad$ Create the state $x_2^p = Lsp(x_2)$ in $Pdg(\mathcal{Y})$
9 $\quad\quad\quad\quad$ Insert the transition $\langle (x^p, x), (o, \mathcal{L}(x), f), (x_2^p, x_2)\rangle$ into $Pdg(\mathcal{Y})$
10 $\quad\quad$ **foreach** *arc* $\langle (x^p, x), (o, r, f), (x_2^p, x_2)\rangle$ *in* τ **do**
11 $\quad\quad\quad r_2 \leftarrow r^p r f$
12 $\quad\quad\quad$ **if** $(x_2^p, r_2^p) \in \mathbb{C}_{\text{new}}$ **then**
13 $\quad\quad\quad\quad$ Substitute $\left(x_2^p, \left(r_2^p | r_2\right)\right)$ for $\left(x_2^p, r_2^p\right)$ in $\mathbb{C}_{\text{new}}$
14 $\quad\quad\quad$ **else**
15 $\quad\quad\quad\quad$ Insert (x_2^p, r_2) into $\mathbb{C}_{\text{new}}$
16 $\quad \mathbb{C} \leftarrow \mathbb{C}_{\text{new}}$
17 Remove from $\mathbb{C}$ every context (x^p, r^p) where x^p does not include any quiescent state
18 **if** $\mathbb{C} = \{(x^p, r^p)\}$ **then**
19 $\quad \mathcal{R} \leftarrow r^p \mathcal{L}(x^p)$
20 **else if** $\mathbb{C} = \left\{\left(x_1^p, r_1^p\right), \ldots, \left(x_k^p, r_k^p\right)\right\}$ *where* $k > 1$ **then**
21 $\quad \mathcal{R} \leftarrow \left(r_1^p \left(\mathcal{L}\left(x_1^p\right)\right)\right) \mid \ldots \mid \left(r_k^p \left(\mathcal{L}\left(x_k^p\right)\right)\right)$.

diagnoser $Pdg(\mathcal{Y})$ instead of $Tdg(\mathcal{Y})$. Second, and more to the point, some new lines of its pseudocode (lines 5–9) are meant to upgrade $Pdg(\mathcal{Y})$ in order to add the possibly missing transitions inherent to the observation o. Lines 10–21 in HYBRID CANDIDATES are the same as lines 5–16 in FAST CANDIDATES. At the end of the execution, $Pdg(\mathcal{Y})$ has possibly been extended based on the temporal observation $\mathcal{O}$.

Example 15 Consider the partial temporal diagnoser $Pdg(\mathcal{W})$ displayed on top of Fig. 14.10. Outlined in Table 14.4 is the execution of the HYBRID CANDIDATES algorithm applied on the temporal observation $\mathcal{O} = [po, bo, po]$. Specifically, for each observation in $\mathcal{O}$, the set $\mathbb{C}$ of contexts and the possibly created states in $Pdg(\mathcal{W})$ are indicated. At the beginning, $\mathbb{C}$ is the singleton $\{(\mathbf{0}, \varepsilon)\}$. With the first observation po, since there is an arc $\langle (\mathbf{0}, (po, \varepsilon, \varepsilon), \mathbf{1}\rangle$ in $Pdg(\mathcal{W})$ already, the computation continues at line 10, thereby assigning $\mathbb{C}_{\text{new}}$ with the singleton $\{(\mathbf{1}, \varepsilon)\}$, as shown in the second row of Table 14.4. With the second observation, namely bo (third row in Table 14.4), since the state $\mathbf{1}$ has no exiting arc marked with a triple involving the observation

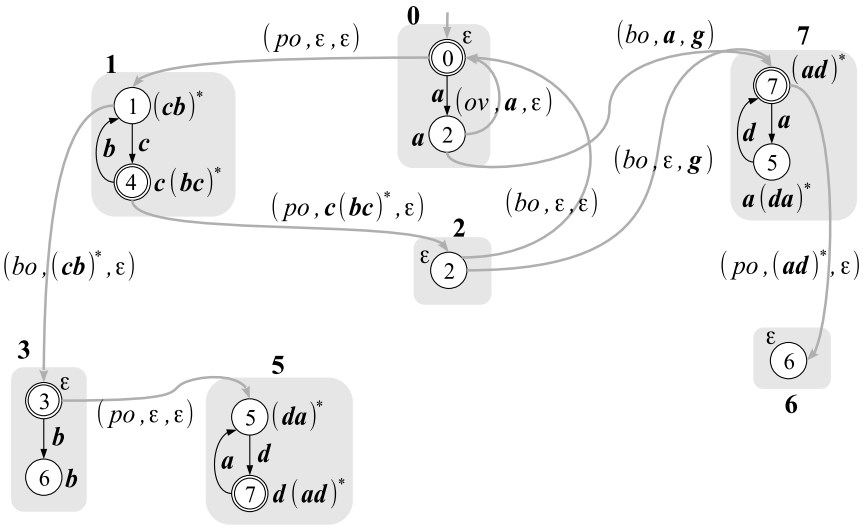

Fig. 14.10 Extension of the partial temporal diagnoser of DES $\mathcal{W}$ (states **0**, **1**, **2**, **6** and **7**) with the new states **3** and **5** and the inherent transitions, by the HYBRID CANDIDATES algorithm, based on the temporal observation $\mathcal{O} = [po, bo, po]$

bo (line 5), the new state **3** is generated as the local space *Lsp*(3), along with a new transition $\langle(\mathbf{1}, 1)), (bo, (\boldsymbol{cb})^*, \varepsilon), (\mathbf{3}, 3)\rangle$. Then, based on lines 10–15, the set $\mathbb{C}$ becomes the singleton $\{(\mathbf{3}, (\boldsymbol{cb})^*)\}$. Next, with the last observation *po* (last row in Table 14.4), since the state 3 has no exiting transitions marked with a triple involving the observation *po*, a new state $\mathbf{5} = Lsp(5)$ is created along with a new transition $\langle(\mathbf{3}, 3)), (po, \varepsilon, \varepsilon), (\mathbf{5}, 5)\rangle$, with $\mathbb{C}$ being instantiated with $\{(\mathbf{5}, (\boldsymbol{cb})^*)\}$. Eventually (lines 17–21), since the local space **5** includes the quiescent state 6 of $\mathcal{W}$, the (only) context $(\mathbf{5}, (\boldsymbol{cb})^*)$ is preserved in $\mathbb{C}$. Hence, based on line 19, the regular expression specifying the candidate set $\Delta(\mathcal{O})$ will be

$$\mathcal{R} = (\boldsymbol{cb})^* \underbrace{\boldsymbol{d}\,(\boldsymbol{ad})^*}_{\mathcal{L}(\mathbf{5})}.$$

The new portion of the resulting (upgraded) partial temporal diagnoser $Pdg(\mathcal{W})$ is pictorially displayed on the bottom of Fig. 14.10, which comprehends states **3** and **5**, along with two transitions, namely $\langle(\mathbf{1}, 1), (bo, (\boldsymbol{cb})^*, \varepsilon), (\mathbf{3}, 3)\rangle$ and $\langle(\mathbf{3}, 3), (po, \varepsilon, \varepsilon), (\mathbf{5}, 5)\rangle$.

Table 14.4 Tracing of the HYBRID CANDIDATES algorithm based on $\mathcal{O} = [po, bo, po]$ and the partial temporal diagnoser $Pdg(\mathcal{W})$ depicted on top of Fig. 14.10

i	o_i	$\mathcal{O}_i$	States created	$\mathbb{C}$
0		[ ]		$\{(\mathbf{0}, \varepsilon)\}$
1	po	$[po]$		$\{(\mathbf{1}, \varepsilon)\}$
2	bo	$[po, bo]$	$\{\mathbf{3}\}$	$\{(\mathbf{3}, (\boldsymbol{cb})^*)\}$
3	po	$[po, bo, po]$	$\{\mathbf{5}\}$	$\{(\mathbf{5}, (\boldsymbol{cb})^*)\}$

14.5 Implementation of the Diagnosis Engines

The diagnosis engines were coded in the *C* programming language, which was chosen for it is highly flexible and, by means of pointers, it enables an explicit dynamic space allocation, reallocation, and deallocation of the RAM. Thanks to these properties, which are not exhibited by higher-level languages, such as *Java*, the programmer can directly manage the memory consumption so as the data structures can be built and handled more efficiently. Given the purposes of the research described in this chapter, data structures were designed very carefully, since they have a substantial impact on the expenditure of computational resources.

In addition, the developed engines can generate several graphic representations, such as that of a DES, a space, an $\mathcal{O}$ space, and a temporal diagnoser (including a partial one). These visualizations are rendered by the *Graphviz* package[6] along with the *dot* layout engine.

14.5.1 Processing of Regular Expressions

The output of the temporal-oriented diagnosis of a DES is a regular expression, which can more easily be grasped by the user the simpler and shorter it is. In order to reduce the redundancy in the regular expression, which is bound to obscure the interpretation of the diagnosis results, some simplification rules were defined and applied.[7] A regular expression is basically a character string; however, this structure was endowed with some attributes to facilitate the simplification of the regular expression resulting from the composition of some regular expressions. Examples of simplifications rules are listed below (where characters r, p, and q denote regular expressions).

1. If $r = p$, then the alternative of r and p is simplified to r, namely $r \mid p \rightsquigarrow r$.
2. When regular expressions are concatenated, possible simplifications are:

[6] *Graphviz* is the abbreviation for the open source *Graph Visualization Software*, which can be found at `graphviz.org`.

[7] To our knowledge, no standardized technique for simplifying regular expressions is offered in the literature.

$$rr^* \rightsquigarrow r^+ \qquad r^+r^* \rightsquigarrow r^+ \qquad r^+r^+ \rightsquigarrow r^+ \qquad r^*r^* \rightsquigarrow r^*$$

3. When the alternative of regular expressions comes into play, attention is paid in order to avoid including the same regular expression twice. Some relevant simplifications in this direction are:

$$r? \mid r \rightsquigarrow r? \qquad r^* \mid r \rightsquigarrow r^* \qquad r^+ \mid r \rightsquigarrow r^+$$

   Furthermore, parentheses are inserted only when an element is not parenthesized already. For instance, if p is parenthesized, then we can write $p \mid (r)$ rather than $(p) \mid (r)$, or $p? \mid (r)$ rather than $(p)? \mid (r)$.
4. For regular expressions relevant to the transformation involved in algorithms like CANDIDATES or FAULT SPACE, where a state is eliminated along with its entering/exiting transitions, the simplification rules are numerous and similar to those introduced above, such as avoiding unnecessary parentheses or, when identical regular expressions are involved, rewriting the regular expressions as:

$$r(r)^*p \rightsquigarrow (r)^+p \qquad r(p?)^*q \rightsquigarrow rp^*q \qquad r(r^*)^*p \rightsquigarrow r^+p$$

To improve both efficiency and parallelism, no external buffers are used in the manipulation of regular expressions, which, whenever possible, are built in already allocated blocks of memory, thus decreasing the number of memory allocation requests and of copying operations as well.

Parallel code was adopted for several functionalities, even for those that are performed offline, such as the generation of the temporal diagnosers, which includes the regular expression construction. This is the reason for no external buffer is used in handling regular expressions. Also the rendering carried out by *Graphviz* benefits from parallel code execution.

14.5.2 Diagnosis Engines Compared

The performances of the three different engines for a posteriori diagnosis of temporal faults, namely *interpreted*, *compiled*, and *hybrid*, were empirically compared. A personal computer equipped with an Intel Xeon Gold 6140 (1–7 cores) CPU and a 128 GB RAM was engaged for running the experiments, whose aim was to find out how the processing time taken online by each engine to generate the temporal-oriented diagnosis grows with the length of the temporal observation in input. Hence, 15 test cases (i.e. instances of the a posteriori diagnosis problem) were created, where the DES, say D, is the same in all the test cases, whereas the temporal observation has a length that is increased by one for each new test case, starting from a length value equal to 1.

DES D consists of 4 components and 8 links; its number of states per component is 2–3 while its number of transitions per component is 3–6. Only one transition

Table 14.5 Processing time (in sec.) relevant to the three techniques for diagnosis of temporal faults, namely, *interpreted*, *compiled*, and *hybrid*

Interpreted	Compiled	Hybrid
0.014088	0.000099	14.541712
0.05389	0.00165	12.193578
0.355806	0.008831	0.008479
1.464485	0.010589	0.010056
3.470084	0.010537	0.009637
3.788433	0.009988	0.009503
6.457794	0.010587	0.009937
10.117073	0.010089	0.009808
5.522604	0.009827	0.009802
5.178329	0.010063	0.009727
9.834129	0.009604	0.01
10.11738	0.009577	0.01
29.515572	0.009471	0.009919
28.923592	0.009413	0.009768
25.36454	0.009358	0.010134

in D is observable. The resulting DES space has a huge number of states and few observable transitions; therefore, the diagnosis task has to cope with the criticality of a temporal diagnoser that includes many large states. The offline generation of the temporal diagnoser of D, which is needed by the compiled diagnosis engine, took 38.63 s. Notice that, since D has just one observable label, the temporal observation of length $i \in [1 \ldots 15]$ consists in i repetitions of this label.

Each of the 15 instances was processed by each of the three diagnosis engines. Numerical results are shown in Table 14.5, where the i-th row is relevant to the instance whose temporal observation has length i. Each row i, excluding $i = 1$, does not report the total time taken to process the relevant instance; instead, it reports the increment of the total time taken to process such instance with respect to the total time needed to process the instance relevant to row $(i - 1)$. In other words, each row records how much more time is taken by each engine in order to process one more observable label in the temporal observation.

A bar-graph representation of the data listed in Table 14.5 is displayed in Fig. 14.11, where a logarithmic scale is used for the CPU time.

The best online performance is provided by compiled diagnosis, thereby, based on a complete temporal diagnoser. The time grows only initially, then it becomes substantially constant, which denotes a linear growth of the total CPU time with the length of the temporal observation. This online performance is grounded, however, on a total (offline) knowledge compilation. Based on a series of experiments (not reported in this chapter), the computational complexity of the generation of the temporal diagnoser is exponentially increasing with the number of components.

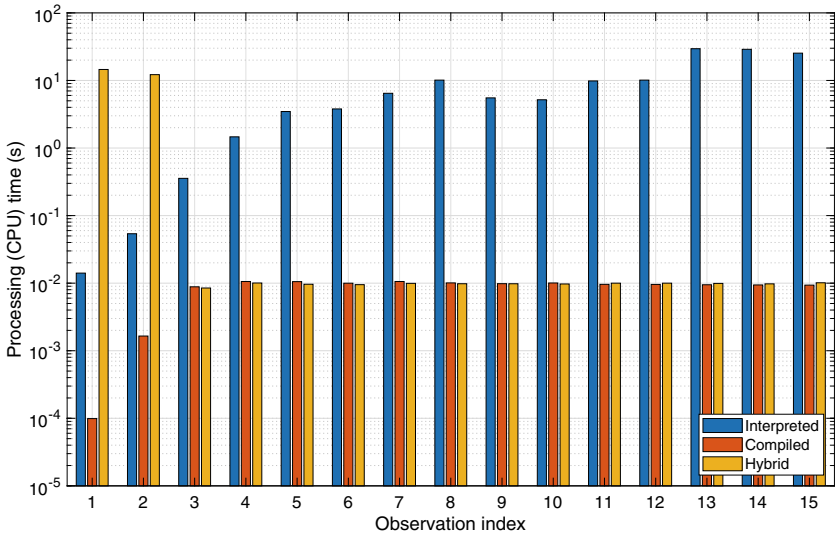

Fig. 14.11 Outline of the processing times of the three diagnosis techniques listed in Table 14.5

Hence, materializing the whole temporal diagnoser is impractical for real DESs. It suffices to consider that the software system developed was unable to generate the complete temporal diagnoser for DESs containing 15 small components.

With hybrid diagnosis, the temporal diagnoser includes initially one state only, without transitions. Consequently, the CPU time necessary to process the first observable labels in the temporal observation is considerably high. On the other hand, with DES D, the hybrid technique succeeds in completing the construction of the temporal diagnoser by processing two observations; afterwards, it basically exhibits the same performance as the compiled technique. In particular, the first observation allows for the creation of 42 states and 1123 transitions, while, after the second observation, the number of states and transitions become 49 and 1395, respectively, thereby completing the generation of the temporal diagnoser. We conjecture that this *modus operandi* of the hybrid technique can be generalized to DESs larger than the DES D: the time for processing each additional observation progressively decreases to a point in which it remains almost constant, where the actual number of observations in the transient depends on the size of the DES.

The performance of the interpreted technique, which is initially good because of the simplicity of its implementation, invariably declines in a few observations, indicating that interpreted diagnosis may be adopted when the temporal observation is short. Even worse, computing the $\mathcal{O}$ spaces of a DES with low observability may become impractical, which makes interpreted diagnosis ineffective.

14.6 Conclusion

This chapter deals with the decision support provided by a model-based approach to diagnosis of partially-observable dynamical systems represented as DESs. A (distributed) DES consists of several components, each endowed with an internal behavior that is represented explicitly as a communicating automaton, while no global behavioral model of the DES is built: component models can be processed upfront to compile knowledge structures that can speed up the diagnostic reasoning. The diagnosis task is performed *a posteriori*, once a chronological sequence of observable events relevant to the system being operated has been recorded. This *temporal observation* is the input of the diagnosis task, while the output amounts to a regular language cumulatively representing all the alternative sequences of faults that can have produced it. Each sequence of faults, or *temporal fault*, is temporally ordered. This *temporal-oriented* diagnosis output, which differs from the classical set-oriented diagnosis output of most approaches in the literature, may be helpful for supporting decision-making when the system comes to an abrupt halt and is no longer being operated. In fact, a human operator can analyze the computed regular language, which concisely represents all the alternative hypotheses about what has possibly occurred in the system, and therefore decide what to do next.

This scenario raises several concerns, the first being the performance of the approach that computes the diagnosis results, as the number of states belonging to the DES behavioral space blows up exponentially. This computational issue is faced in the present chapter, where three different techniques to compute the temporal-oriented diagnosis output are investigated, namely *interpreted*, *compiled*, and *hybrid* diagnosis. Empirical evidence (cf. Sect. 14.5.2) confirms that both *interpreted* and *compiled* diagnosis are adequate for DESs including few components only. The *hybridization* of the *interpreted* and *compiled* diagnosis approaches can instead provide some complexity mitigation.

A second concern is about the comprehensibility of the regular language that is produced as an output by the diagnosis task. Although the redundancy in the resulting regular expression has been reduced by means of a variety of simplification rules (cf. Sect. 14.5.1), one may object that such an expression is not easy to understand for a diagnostician who has no background in computer science. The cognitive understanding of regular expressions by users with no background knowledge, after the topic has been shortly introduced, could be studied in order to reply to this argument. Multiple representations of regular expressions could be envisaged, thus making the study comparative. Some suggestions about how to present the diagnosis output to human users could be drawn from the literature about *explainable AI*, a branch of research that, as illustrated in [33], is situated in the human-agent interaction area. However, these investigations are beyond the scope of the present chapter.

A third concern is inherent to the cognitive load that may affect the human operator in case the regular language output by the diagnosis engine encompasses a high or even unbounded number of alternative temporal faults. The number of computed alternatives could be reduced if some preference criteria and/or some feasibility

constraints were used as a filter. This filter could be applied while postprocessing the diagnosis result, in other words, the domain-agnostic approach described in this chapter could be coupled with a domain-specific postprocessor. Another way to decrease the number of alternative sequences of faults in the diagnosis result could be by empowering the approach described in this chapter so as it can natively manage some additional knowledge about the preference criteria and/or the feasibility constraints. This way, the approach could produce all and only the preferred candidates that comply with the given feasibility constraints. For instance, a preference criterion could be interested just in sequences including three fault occurrences at most. A feasibility constraint, for instance, could be relevant to the time taken by the trajectories (belonging to the global DES behavioral model) that produce the given temporal observation: a trajectory is feasible only under the condition that it takes a time compatible with the timespan in which the temporal observation was collected. This filter would rule out possible trajectories that produce the given temporal observation but that include more than three fault occurrences and/or take a time length that is not compliant with the known time length of the observation interval. A tighter constraint could impose every feasible trajectory to produce each observable event in the temporal observation exactly at the time instant when it was recorded. Accounting for the mentioned feasibility constraints, however, would require knowing the time taken by each transition in the DES behavioral space. So far, only quite abstract untimed models of the DES components have been adopted in the approach described. Hence, there is much room for future investigation about preferences and feasibility constraints.

Another concern is relevant to the possible enhancement of the support provided by the DSS described in the previous sections. The DSS produces as output a regular language where each symbol is a fault. The output could possibly be more helpful if the DSS would automatically draw the faults that have occurred for sure since they are present in all the strings of the regular language. This improvement can be attained quite easily by a postprocessor. Further help could possibly be provided by computing the *health state* [42] of the DES, that is, the posterior probability of each component to be affected by each specific fault, under the assumption that the prior probabilities of individual faults are known. Although this assumption has not been made for the DESs considered in this chapter, conceptually, the models of DES components can embed prior fault probabilities. In [42], set-oriented candidates are exploited, in an adaptive number, to estimate the health state. Drawing the health state of a DES from (an adaptive number of) temporal faults is a subject for future research. Temporal faults could support the estimation of the likelihood that a certain fault pattern and/or some intermittent faults have occurred. Another research could focus on automatically assessing the criticality (e.g. from the safety point of view) of the diagnosis result. The knowledge about the (domain-dependent) criticality could be embedded in some (possibly annotated) behavioral scenarios (cf. Sect. 14.4.3) to be exploited by a (domain-independent) postprocessor.

The DSS described in this chapter produces as output some results that have to be analyzed by a human expert in order to take a decision. A final concern about the DSS is whether it could be extended to offer some assistance to the human operator

in choosing what to do next. For instance, based on the DES component models, the DSS could suggest probing the real system in order to gather a new observable event, or performing a maintenance/repair action that is an observable event encompassed by the DES component models. In either case, the a posteriori diagnosis engine can be run anew in order to take as input the former temporal observation extended with the new observable event. In suggesting the most appropriate observable event to be taken, the DSS should be guided by a predefined (possibly multiple) goal, such as, minimizing the number or cost or time of the additional observations that are needed to clarify what has happened in the system. An alternative suggestion that could be given by the DSS is limiting the considered system in its operation from now on. For instance, if a fault is certain, instead of repairing it, it may be convenient to drive the system so as its operation does not involve the faulty component (or its faulty part) any longer.

Acknowledgements This work was supported by the EU NEXTGENERATIONEU program within the PNRR Future Artificial Intelligence - FAIR project (PE0000013, CUP H23C22000860006), Objective 10: Abstract Argumentation for Knowledge Representation and Reasoning, specifically by the project Argumentation for Informed Decisions with Applications to Energy Consumption in Computing—AIDECC (CUP D53C24000530001).

References

1. F. Basile, Overview of fault diagnosis methods based on Petri net models, in *Proceedings of the 2014 European Control Conference, ECC 2014* (2014), pp. 2636–2642. https://doi.org/10.1109/ECC.2014.6862631
2. A. Benveniste, E. Fabre, S. Haar, C. Jard, Diagnosis of asynchronous discrete-event systems: A net unfolding approach. IEEE Trans. Autom. Control **48**, 714–727 (2003)
3. M. Bertl, P. Ross, D. Draheim, A survey on AI and decision support systems in psychiatry - uncovering a dilemma. Expert Syst. Appl. **202**, 117464 (2022)
4. N. Bertoglio, G. Lamperti, M. Zanella, X. Zhao, Diagnosis of temporal faults in discrete-event systems, in *24th European Conference on Artificial Intelligence (ECAI 2020), Frontiers in Artificial Intelligence and Applications*, ed, by G.D. Giacomo, A. Catala, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang, vol. 325, pp. 632–639 (IOS Press, Amsterdam, 2020). https://doi.org/10.3233/FAIA200148
5. N. Bertoglio, G. Lamperti, M. Zanella, X. Zhao, Explanatory diagnosis of discrete-event systems with temporal information and smart knowledge-compilation, in *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, ed. by D. Calvanese, E. Erdem, M. Thielsher (IJCAI Organization, 2020), pp. 130–140. https://doi.org/10.24963/kr.2020/14
6. N. Bertoglio, G. Lamperti, M. Zanella, X. Zhao, Explanatory monitoring of discrete-event systems, in *Intelligent Decision Technologies 2020*, *Smart Innovation, Systems and Technologies*, ed. by I. Czarnowski, R. Howlett, L. Jain, vol. 193 (Springer, Singapore, 2020), pp. 63–77. https://doi.org/10.1007/978-981-15-5925-9_6
7. N. Bertoglio, G. Lamperti, M. Zanella, X. Zhao, Temporal-fault diagnosis for critical-decision making in discrete-event systems, in *Knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 24th International Conference KES2020*, *Procedia Computer Science*, ed. by M. Cristani, C. Toro, C. Zanni-Merk, R. Howlett, L. Jain, vol. 176 (Elsevier, 2020), pp. 521–530.https://doi.org/10.1016/j.procs.2020.08.054

8. D. Brand, P. Zafiropulo, On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983). https://doi.org/10.1145/322374.322380
9. J. Brzozowski, E. McCluskey, Signal flow graph techniques for sequential circuit state diagrams. IEEE Trans. Electron. Comput. **EC-12**(2), 67–76 (1963)
10. M.P. Cabasino, A. Giua, C. Seatzu, Fault detection for discrete event systems using Petri nets with unobservable transitions. Automatica **46**, 1531–1539 (2010)
11. C. Cassandras, S. Lafortune, *Introduction to Discrete Event Systems*, 2nd edn. (Springer, New York, 2008)
12. C. Ching-Chin, A.I. Ka Ieng, W. Ling-Ling, K. Ling-Chieh, Designing a decision-support system for new product sales forecasting. Expert Syst. Appl. **37**(2), 1654–1665 (2010). https://doi.org/10.1016/j.eswa.2009.06.087.
13. J.M. Choffray, G.L. Lilien, A decision-support system for evaluating sales prospects and launch strategies for new products. Ind. Mark. Manage. **15**(1), 75–85 (1986). https://doi.org/10.1016/0019-8501(86)90046-5
14. X. Cong, M. Fanti, A. Mangini, Z. Li, Decentralized diagnosis by Petri nets and integer linear programming. IEEE Trans. Syst., Man, Cybern.: Syst. **48**(10), 1689–1700 (2018)
15. B. Dowdeswell, R. Sinha, S. MacDonell, Finding faults: a scoping study of fault diagnostics for industrial cyber-physical systems. J. Syst. Softw. **168**, 1–16 (2020). https://doi.org/10.1016/j.jss.2020.110638
16. A. Grastien, P. Haslum, *Diagnosis as planning: two case studies, in Scheduling and Planning Applications Workshop (SPARK 2011)* (Freiburg, Germany, 2011), pp.37–44
17. A. Grastien, P. Haslum, S. Thiébaux, Conflict-based diagnosis of discrete event systems: theory and practice, in *Thirteenth International Conference on Knowledge Representation and Reasoning (KR 2012)* (Association for the Advancement of Artificial Intelligence, Rome, 2012), pp. 489–499
18. R.A. Guimapi, S.A. Mohamed, L. Biber-Freudenberger, W. Mwangi, S. Ekesi, C. Borgemeister, H.E.Z. Tonnang, Decision support system for fitting and mapping nonlinear functions with application to insect pest management in the biological control context. Algorithms **13**(4) (2020). https://doi.org/10.3390/a13040104. https://www.mdpi.com/1999-4893/13/4/104
19. W. Hamscher, L. Console, J. de Kleer (eds.), *Readings in Model-Based Diagnosis* (Morgan Kaufmann, San Mateo, 1992)
20. G. Jiroveanu, R. Boel, B. Bordbar, On-line monitoring of large Petri net models under partial observation. J. Discrete Event Dyn. Syst. **18**, 323–354 (2008)
21. M. Khakifirooz, M. Fathi, P.M. Pardalos, D.J. Power, Decision support for smart manufacturing, in *Encyclopedia of Organizational Knowledge, Administration, and Technology*, ed. by M. Khosrow-Pour (IGI Global, 2021), pp. 2352–2364. https://doi.org/10.4018/978-1-7998-3473-1.ch162
22. F. Khemakhem, H. Ellouzi, H. Ltifi, M.B. Ayed, Agent-based intelligent decision support systems: a systematic review. IEEE Trans. Cognit. Develop. Syst. **14**(1), 20–34 (2022). https://doi.org/10.1109/TCDS.2020.3030571
23. J. de Kleer, B. Williams, Diagnosing multiple faults. Artif. Intell. **32**(1), 97–130 (1987)
24. E. Koukoutsis, C. Papaodysseus, G. Tsavdaridis, N.V. Karadimas, A. Ballis, E. Mamatsi, A.R. Mamatsis, Design limitations, errors and hazards in creating decision support platforms with large- and very large-scale data and program cores. Algorithms **13**(12) (2020). https://doi.org/10.3390/a13120341. https://www.mdpi.com/1999-4893/13/12/341
25. G. Lamperti, S. Trerotola, M. Zanella, X. Zhao, Sequence-oriented diagnosis of discrete-event systems. J. Artif. Intell. Res. **78**, 69–141 (2023). https://doi.org/10.1613/jair.1.14630
26. G. Lamperti, M. Zanella, Monitoring of active systems with stratified uncertain observations. IEEE Trans. Syst., Man, Cybern. - Part A: Syst. Humans **41**(2), 356–369 (2011). https://doi.org/10.1109/TSMCA.2010.2069096
27. G. Lamperti, M. Zanella, X. Zhao, Introduction to Diagnosis of Active Systems (Springer. Cham (2018). https://doi.org/10.1007/978-3-319-92733-6
28. B. Li, M. Khlif-Bouassida, A. Toguyéni, Reduction rules for diagnosability analysis of complex systems modeled by labeled Petri nets. IEEE Trans. Autom. Sci. Engin. (2019). https://doi.org/10.1109/TASE.2019.2933230

29. J. Lunze, Diagnosis of quantized systems based on a timed discrete-event model. IEEE Trans. Syst., Man, Cybern. - Part A: Syst. Humans **30**(3), 322–335 (2000)
30. C. Lv, Research on intelligent decision support system for automobile fault diagnosis based on SWOT analysis, in *2021 2nd International Conference on Artificial Intelligence and Information Systems, ICAIIS 2021* (Association for Computing Machinery, New York, 2021). https://doi.org/10.1145/3469213.3471312. https://doi.org/10.1145/3469213.3471312
31. López-Martínez, F., Núñez-Valdez, E.R., García-Díaz, V., Bursac, Z.: A case study for a big data and machine learning platform to improve medical decision support in population health management. Algorithms **13**(4) (2020https://doi.org/10.3390/a13040102. https://www.mdpi.com/1999-4893/13/4/102
32. S. McIlraith, Explanatory diagnosis: conjecturing actions to explain observations, in *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR 1998)* (Morgan Kaufmann, S. Francisco, CA, Trento, I, 1998), pp. 167–177
33. T. Miller, Explanation in artificial intelligence: insights from the social sciences. Artif. Intell. **267**, 1–38 (2019)
34. Y. Pencolé, M. Cordier, A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks. Artif. Intell. **164**(1–2), 121–170 (2005)
35. Y. Pencolé, G. Steinbauer, C. Mühlbacher, L. Travé-Massuyès, Diagnosing discrete event systems using nominal models only, in *28th International Workshop on Principles of Diagnosis (DX'17)*, ed. by M. Zanella, I. Pill, A. Cimatti, vol. 4 (Kalpa Publications in Computing, 2018), pp. 169–183. https://doi.org/10.29007/1d2x
36. F. Rajaei, S. Cheng, C.A. Williamson, E. Wittrup, K. Najarian, AI-based decision support system for traumatic brain injury: a survey. Diagnostics **13**(9) (2023). https://doi.org/10.3390/diagnostics13091640. https://www.mdpi.com/2075-4418/13/9/1640
37. N. Ran, H. Su, A. Giua, C. Seatzu, Codiagnosability analysis of bounded Petri nets. IEEE Trans. Autom. Control **63**(4), 1192–1199 (2018)
38. R. Reiter, A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
39. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Diagnosability of discrete-event systems. IEEE Trans. Autom. Control **40**(9), 1555–1575 (1995)
40. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Failure diagnosis using discrete-event models. IEEE Trans. Control Syst. Technol. **4**(2), 105–124 (1996)
41. E. Shortliffe, *Computer-Based Medical Consultations: MYCIN* (American Elsevier, New York, 1976)
42. R. Stern, M. Kalech, S. Rogov, A. Feldman, How many diagnoses do we need? Artif. Intell. **248**, 26–45 (2017)
43. P. Struss, Fundamentals of model-based diagnosis of dynamic systems, in *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 1997)* (Nagoya, Japan, 1997), pp. 480–485
44. V. Uraikul, C.W. Chan, P. Tontiwachwuthikul, Artificial intelligence for monitoring and supervisory control of process systems. Eng. Appl. Artif. Intell. **20**(2), 115–131 (2007). https://doi.org/10.1016/j.engappai.2006.07.002
45. X. Yin, S. Lafortune, On the decidability and complexity of diagnosability for labeled Petri nets. IEEE Trans. Autom. Control **62**(11), 5931–5938 (2017)

Author Biographies

Gianfranco Lamperti received the Doctoral degree in Electronics Engineering (Computer Science) from Politecnico di Milano, Italy, in 1986. After working in the private sector in several research and development projects, he joined the University of Brescia, Italy, in 1995 as an Assistant Professor. He is currently an Associate Professor of Computer Science with the Department of Information Engineering, University of Brescia. His current research interests include engineering issues in diagnosis of discrete-event systems, uncertainty, and processing of finite automatons.

Stefano Trerotola a distinguished graduate in Computer Science and Engineering in 2022 at the University of Brescia, Italy, has long been interested into computer science, technology, and logic. Currently employed as a back-end developer at a forefront Field Service Management company, he works on daily challenges to ensure the effective functioning of the systems he develops.

Marina Zanella received the Doctoral degree in Electronics Engineering (Computer Science) from Politecnico di Milano, Italy, in 1986. She is currently an Associate Professor of Computer Science with the Department of Information Engineering, University of Brescia, Italy. Her research interests include knowledge-based systems, diagnosability analysis, model-based reasoning for monitoring and diagnosis of static systems and discrete-event systems.