Diagnosis and diagnosability analysis using PEPA

Luca Console, Claudia Picardi, Marina Ribaudo[1]

Abstract. In this paper we propose the use of process algebras as powerful frameworks for model-based diagnosis. In fact, they provide machinery and tools for building component-oriented models, for characterizing and computing diagnoses, and for analyzing properties such as the diagnosability of the system under investigation.

1 Introduction

The definition of frameworks for characterizing diagnosis attracted a lot of attention in the model-based reasoning community (see e.g., chapt. 2 of [7] or [2, 5]). These frameworks have been used to provide a semantics for diagnostic problem solving and to analyze its properties (e.g., its computational complexity). Moreover, they have been used to analyze properties of the system to be diagnosed, e.g., studying the suitability of different types of models [21] or sensor placement for diagnosability [20] or, more generally, diagnosability [18].

Although logic is the formalism most frequently used in AI frameworks for diagnosis [15, 14, 4], the comparison with other formalisms such as FDI techniques [12] or discrete event systems [17, 16], is attracting increasing attention. Indeed, there is a tradition in the use of discrete event systems for modeling physical systems and for analyzing their functional properties [11, 13], their performance [8, 1], reliability and fault tolerance [10]. However, only in a few cases these approaches have been used in the context of diagnosis [18, 16].

In this paper we propose the use of process algebras for modeling physical systems, characterizing their diagnoses and analyzing their diagnostic properties. Process algebras (and, specifically PEPA – Performance Evaluation Process Algebra [8]) are expressive languages which enable the high-level definition of complex systems, supporting a component centered modeling paradigm. Equivalence notions, which can be used to compare models, have been formally defined and some extensions of the formalism include sophisticated models of time (time, however, is out of the scope of this preliminary work). Moreover, tools are available for model specification and analysis (both qualitative and quantitative).

In the paper, after a brief sketch of PEPA (Sec.3), we show how this algebra supports the modeling of complex physical systems (Sec.4) and the characterization of diagnosis (Sec.5). In Sec. 6 we show how the formalism can be applied to investigating diagnosability of a physical system, given the available sensors and the assumptions made in the model. Finally, Sec. 7 provides conclusions and comparisons.

[1] Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy, Phone (+39) 011 6706711, e-mail: {lconsole,picardi,marina}@di.unito.it

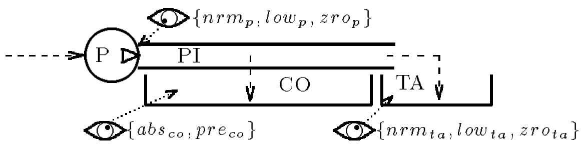

Figure 1. Abstract representation of the running example

2 A running example

In this section we introduce a simple device (see Fig.1) that we shall use as a running example throughout the paper. It is formed by a pump P which delivers water to a tank TA via a pipe PI; another tank CO is used as a collector for water that may leak from the pipe. For the sake of simplicity we assume that the pump is always on and supplied of water.

The pump P has three modes of behavior: ok (the pump produces a normal output flow), leaking (it produces a low output flow) and blocked (no output flow). The pipe PI can be ok (delivering to the tank the water it receives from the pump) or leaking (in this case we assume that it delivers to the tank a low output when receiving a normal or low input, and no output when receiving no input). The tanks TA and CO are always in mode ok, i.e., they simply receive water.

We assume that three sensors are available (see the eyes in Fig.1): $flow_P$ measures the flow from the pump, which can be normal (nrm_p), low (low_p), or zero (zro_p); $level_{TA}$ measures the level of the water in TA, which can be normal (nrm_{ta}), low (low_{ta}), or zero (zro_{ta}); $level_{CO}$ records the presence of water in CO, either present (pre_{co}) or absent (abs_{co}).

A PEPA model of this device is presented in Sec.3. For the sake of simplicity we start by abstracting dynamics; we then show how dynamic behavior can be naturally modeled. In particular, we first consider a stateless abstraction in which the level of water in the tank depends only on the input to the tank; we then discuss a more accurate model in which the level is a "state variable" whose change depends on the input. On the other hand, we regard time as a sequence of states (events), without any metric (see the discussion in [2]).

3 PEPA

In this section we briefly introduce the syntax of Performance Evaluation Process Algebra (PEPA) [8], an algebraic description technique based on classical process algebras [9, 11] and enhanced with timing information. This extension results in models which can be used to calculate performance measures as well as deduce functional properties of a system.

PEPA supports a compositional approach to modeling: the component types occurring in a system are modeled in isolation; then the structure of the system is described as a composition of component instances.

The components execute *activities* $a = (\alpha, r)$. Each activity is represented by two pieces of information: the label or *action type*, $\alpha \in \mathcal{A}$, which identifies it, and the *activity rate* r which is the parameter of the negative exponential distribution function expressing its duration. The set of possible action types includes a distinguished type, τ, which denotes internal details and provides an important abstraction mechanism. The restriction to negative exponential distributions allows the derivation of Markov processes and the computation of performance measures starting from PEPA models. However, since timing information is not considered in this paper, we will omit action rates—and their implications—in the following discussion.

PEPA uses a small set of operators for the specification of language expressions[2]. The grammar of the language is shown below, the names and the meanings of the operators are informally presented referring to the example of Sec.2.

$$
\begin{array}{lll}
S & ::= \ A \mid (\alpha, r).S \mid S + S & (\textit{components behavior}) \\
P & ::= \ S \mid P \underset{L}{\bowtie} P \mid \ P/H & (\textit{system structure})
\end{array}
$$

The operators in the definition of S can be used to define component types behaviors; those in the definition of P allow the description of the structure of a system.

Constant. The equation $S \stackrel{def}{=} A$ gives the constant S the behavior of component A. In this way we assign names to components and we can define recursive behaviors.

Prefix. This is the basic mechanism for specifying behaviors: $\alpha.S$ carries out activity α and subsequently behaves as S. Sequences of actions can be combined to build complex terms. For example, the behavior of a pump in mode ok could be $Pok_1 \stackrel{def}{=} nrm_0.nrm_p.P$: after receiving a normal input flow (nrm_0), the pump produces a normal output flow (nrm_p) and then behaves as P. For the leak modality we can write $Plk_1 \stackrel{def}{=} nrm_0.low_p.P$, the pump producing a low output (low_p).

Choice. The choice operator captures the possibility of selection among alternatives. For example, $P \stackrel{def}{=} Pok_1 + Plk_1$ represents a pump which may behave either as Pok_1 *or* as Plk_1. If all the initial activities (preconditions) of Pok_1 and Plk_1 are enabled, the choice is nondeterministic.

Cooperation. The pump P passes water to the pipe PI and this can be modeled in PEPA by means of the cooperation operator, i.e. we can write $P \underset{L}{\bowtie} PI, L = \{nrm_p, low_p\}$. The two components P and PI proceed asynchronously with any action not contained in the *cooperation set* $L = \{nrm_p, low_p\}$. However, for any action in L, they must cooperate. Such action, called *shared* action, is enabled in $P \underset{L}{\bowtie} PI$ when it is enabled in the two individual components. If the cooperation set is empty, the two components proceed independently and we use the compact notation $P \parallel PI$, instead of $P \underset{\emptyset}{\bowtie} PI$. Notice that this type of cooperation is not restricted to pairs of components but it can involve more than two entities.

Hiding. This operator abstracts some aspects of the behavior of a component: only action types relevant for the current analysis are visible to an external observer, or to other components. In the expression P/H, H represents the set of actions that are hidden; for example nrm_0, in $P/\{nrm_0\}$.

[2] All the details can be found in [8] or at the PEPA Web page www.dcs.ed.ac.uk/pepa.

4 Modeling a physical system

In component oriented approaches to model-based diagnosis, the model of a physical system is usually divided in two parts:

1. *A model of the behavior of each component type.* In particular, for each component, we recognize: (a) a set of *behavioral modes*, including the correct ok mode and a set of fault modes; (b) a set of interface variables, corresponding to input/output to/from the component; (c) (possibly) a set of state variables; (d) a set of relations between the variables that describe, for each mode, the behavior of the component. Such a behavior is described in a context independent way, i.e., independently of the role that the instances of the component type may play within the system.
2. *A model of the structure of the system*, i.e., the enumeration of the component instances and their connections.

These types of models can be easily expressed using PEPA. The behavior of each component type in the example of Fig.1 can be described as a nondeterministic choice between the various modes. For each mode, we define a set of equations (Fig.2) that specify the relations between the component variables. In particular, the actions of PEPA are used to express conditions on input, output (and state) variables.

In the following we discuss in detail one component, the pipe PI, the behavior of the other components being similar. First of all, PI may either be in the ok mode ($PIok_1$) or in the leaking mode ($PIlk_1$). This is expressed by the first equation PI: the alternatives within the choice correspond to the modes of behavior of the component. The additional identifier End allows the component to evolve into a final (absorbing) state whose meaning will be clear in Sec.5.

When the ok mode is selected, the subsequent behavior depends on the input to the pipe. In the equation $PIok_1$ we have a choice: for each alternative we have an action that specifies a precondition on the input and, according to its value, the pipe behaves as $PIok_2$, $PIok_3$ or $PIok_4$. In each case the pipe produces two values: one is its output, the other is used to simulate leakage. For example, in $PIok_2$ we have a normal (nrm_1) output and no leakage (abs_2)[3].

The equations describing a component are recursive: in this way we can describe its behavior across time. As we noticed in Sec.2, we consider time as a sequence of states (events); thus a pipe, for example, may receive a stream of inputs across time. In fact, after producing two outputs it starts again behaving as PI, i.e., it can process another input. For each input it can behave either correctly or faulty and therefore also time-varying and intermittent behavior can be easily modeled.

The example of Fig. 1 is composed of a pump, a pipe, a tank, a collector, which are instances of the corresponding component types. We use indexes to denote instances; $P^{(1)} : P$ means that $P^{(1)}$ is an instance of the component type P (for the sake of simplicity we shall omit indexes on the names of the actions). Equation SD_1 describes the structure of this system in terms of its components; the cooperation operator is used to specify their connections and the sets L_i define action types on which the components synchronize. The hiding operator is used to hide non relevant behavior: actions belonging to the set H, representing the input to the pipe and the internal flows to TA and CO, are now invisible.

[3] The subscripts are used to distinguish the two outputs of the pipe.

Components behavior

$$
\begin{array}{llll}
(1) & P & \stackrel{def}{=} & Pok_1 + Plk_1 + Pbl_1 + End \\
 & Pok_1 & \stackrel{def}{=} & nrm_0.nrm_p.P \\
 & Plk_1 & \stackrel{def}{=} & nrm_0.low_p.P \\
 & Pbl_1 & \stackrel{def}{=} & nrm_0.zro_p.P \\
(2) & PI & \stackrel{def}{=} & PIok_1 + PIlk_1 + End \\
 & PIok_1 & \stackrel{def}{=} & nrm_p.PIok_2 + low_p.PIok_3 + zro_p.PIok_4 \\
 & PIok_2 & \stackrel{def}{=} & nrm_1.abs_2.PI \\
 & PIok_3 & \stackrel{def}{=} & low_1.abs_2.PI \\
 & PIok_4 & \stackrel{def}{=} & zro_1.abs_2.PI \\
 & PIlk_1 & \stackrel{def}{=} & nrm_p.PIlk_2 + low_p.PIlk_2 + zro_p.PIlk_3 \\
 & PIlk_2 & \stackrel{def}{=} & low_1.pre_2.PI \\
 & PIlk_3 & \stackrel{def}{=} & zro_1.abs_2.PI \\
(3) & TA & \stackrel{def}{=} & TAok_1 + End \\
 & TAok_1 & \stackrel{def}{=} & nrm_1.nrm_{ta}.TA + low_1.low_{ta}.TA \\
 & & & \qquad +zro_1.zro_{ta}.TA \\
(4) & CO & \stackrel{def}{=} & COok_1 + End \\
 & COok_1 & \stackrel{def}{=} & pre_2.pre_{co}.CO + abs_2.abs_{co}.CO \\
(5) & End & \stackrel{def}{=} & end.End
\end{array}
$$

Structure of the device

component instances

$$P^{(1)} : P;\ PI^{(1)} : PI,\ TA^{(1)} : TA;\ CO^{(1)} : CO$$

connections

$$
\begin{array}{lll}
SD_1 & \stackrel{def}{=} & (P^{(1)} \underset{L_1 \cup \{end\}}{\bowtie} (PI^{(1)} \underset{L_2 \cup \{end\}}{\bowtie} \\
 & & (TA^{(1)} \underset{\{end\}}{\bowtie} CO^{(1)}))) \;/\; H
\end{array}
$$

$L_1 = \{nrm_p, low_p, zro_p\}, L_2 = \{nrm_1, low_1, zro_1, abs_2, pre_2\}$
$H = \{nrm_0, nrm_1, low_1, zro_1, abs_2, pre_2\}$

Figure 2. PEPA model for the example in Fig.1

The semantics of each (untimed) PEPA term is given via a *Labeled Transition System* LTS $= (\mathcal{C}, \mathcal{A}, \xrightarrow{\alpha})$ where $\mathcal{C}$ is the set of states, corresponding to syntactic terms of the language, $\mathcal{A}$ is the set of actions, and the transition relation $\xrightarrow{\alpha}$ specifies which action causes one state to evolve into another.

The LTS describes all possible evolutions of the components, either individually or in cooperation and can be automatically computed using the PEPA Workbench [6], a prototype tool which supports modeling and analysis with PEPA. Qualitative properties, such as absence of deadlocks or the reachability of a given state, can be studied by investigating it. Model-checking techniques [3] can also be used to verify system properties expressed using logical formulas. In the following we will use LTS_{SD} to denote the transition system underlying a physical system SD.

We end this section by briefly discussing an example of equations for describing dynamic behavior. In a more accurate model of a tank we should consider the level of the water inside the tank as a state variable, whose change depends on the balance between inputs to and outputs from the tank (the latter are not present in the simple example we are considering). State variables can be modeled as actions, just as input and output variables. A model of the tank could contain the following equations:

$$
\begin{array}{llll}
(i) & TA_1 & \stackrel{def}{=} & TAok_1 + TAlk_1 + End \\
(ii) & TAok_1 & \stackrel{def}{=} & zro_l.TAok_2 + low_l.TAok_3 + \ldots \\
(iii) & TAok_2 & \stackrel{def}{=} & zro_1.zro_l.TA_1 + \ldots \\
 & \ldots & & \\
(iv) & TA_2 & \stackrel{def}{=} & zro_l.zro_l.TA_2 + low_l.low_l.TA_2 + high_l.high_l.TA_2 \\
(v) & TA & \stackrel{def}{=} & TA_1 \underset{L}{\bowtie} TA_2 \textit{ where } L = \{zro_l, low_l, high_l\}
\end{array}
$$

Actions in the set L denote the level of the water (i.e., the values of the state variable associated with the tank). When the tank is ok ($TAok_1$) and the level is zero (zro_l used as precondition in equation (ii)) then the tank behaves as specified by equation (iii): when it receives no input (zro_1) it remains empty (i.e., the state variable maintains the same value zro_l). In order to model the dynamic behavior of the tank, equation (iv) introduces a dummy component TA_2 which is used in (v) to synchronize the output from the tank to the input to the tank itself (in this sense a state variable is regarded as both an input and an output from the component to itself).

5 A characterization of diagnosis

Diagnosis is performed starting from a set of observations, possibly gathered in different states (snapshots). It consists in finding a mode assignment to each component (in each snapshot) such that the observations are explained.

In terms of PEPA, the observations can be expressed as *new* equations. In principle, any equation can be used; the simpler (and common) case is when we merely have a sequence of the values observed by the sensors. For instance:

1. $Obs_1 \stackrel{def}{=} nrm_p.abs_{co}.nrm_{ta}.end.End$ corresponds to observing one value for each sensor in Fig.1. Action *end* is introduced to specify that no more observations are available. All the components in Fig.2 have an associated *End* behavior that is used to terminate the evolution: the system evolves into a final state when all its components execute the shared action *end*.
2. $Obs_2 \stackrel{def}{=} nrm_p.O_2 + low_p.O_2; O_2 \stackrel{def}{=} abs_{co}.nrm_{ta}.end.End$ corresponds to a case where the reading of the flow sensor from the pipe is *imprecise* and we cannot distinguish if it is normal (nrm_p) or low (low_p).
3. $Obs_3 \stackrel{def}{=} nrm_p.abs_{co}.nrm_{ta}.nrm_p.pre_{co}.low_{ta}.end.End$ corresponds to observing two snapshots, i.e., observations in two different states for the three sensors.

Given a system description SD and an observations Obs, the diagnosis can be characterized starting from the equation

$$Diag \stackrel{def}{=} (Obs \underset{S \cup \{end\}}{\bowtie} SD) \;/\; H$$

in which SD and Obs are synchronized over the set S containing all the actions the sensors can witness, and H is the set of non-observable actions (for readability we will omit the set H in the following equations).

If we compare LTS_{SD} with the LTS underlying *Diag* (LTS_{Diag}) we have that: (1) its size is reduced; (2) only some paths lead the system to the final state; (3) some deadlocks—states from which no transitions are possible—are introduced.

Only the paths leading the system to the state enabling action *end* correspond to evolutions which *explain* the observations and are those of interest for our diagnosis purpose.

Other evolutions, which were possible in SD before synchronization are not allowed anymore since they correspond to behaviors which do not completely explain the observations.

For instance, given the equation SD_1 of Fig.2 and the observations Obs_1 above, we can write

$$Diag_1 \stackrel{def}{=} Obs_1 \underset{S\cup\{end\}}{\bowtie} SD_1$$
$$S = \{nrm_p, low_p, zro_p, abs_{co}, pre_{co}, nrm_{ta}, low_{ta}, zro_{ta}\}$$

According to the observation Obs_1, the first sensor witnesses a normal flow nrm_p from the pump. This observation cuts from LTS_{SD_1} all the paths which start with actions low_p or zro_p since they are not offered by Obs_1. Also the paths in LTS_{SD_1} starting with action nrm_p but expecting pre_{co} are pruned because Obs_1 expects abs_{co}. By continuing this reasoning, the only evolutions compatible with the observations are those that eventually enable action end. These are the paths of interest for the diagnosis. Notice that, as a result of the synchronization between SD_1 and the observations, these are the only paths that do not lead to deadlocks in LTS_{Diag}.

More formally, for each component C_i we denote by $\mathbf{B}_i$ the set of its behavioral modes, identified by the constants appearing on the right hand side of the equation defining C_i. We then consider the equation $Diag \stackrel{def}{=} Obs \underset{S\cup\{end\}}{\bowtie} SD$ and the transition system LTS_{Diag}. Each state in LTS_{Diag} is in the form $\widetilde{O} \parallel \widetilde{C}_1 \parallel \ldots \parallel \widetilde{C}_n$ where $\widetilde{O}$ is the state of the observation and $\widetilde{C}_i$ is the state of component C_i. We call *final state* the state $End \parallel \ldots \parallel End$ ($n+1$ times).

Definition 1 *We say that a path $\sigma = s_1 s_2 \ldots s_n$ in the LTS of $Obs \underset{S\cup\{end\}}{\bowtie} SD$ is* consistent *with observations Obs if it leads from the initial state to the final state.*

From each consistent path σ it is possible to extract a diagnosis. Let us start from the case where the observations correspond to a single snapshot (i.e., at most one reading for each sensor). In this case a diagnosis is simply the set of behavioral modes corresponding to states $\widetilde{C}_i$ in σ.

Definition 2 *Given a consistent path $\sigma = s_1 s_2 \ldots s_n$, a candidate diagnosis $\Delta(\sigma)$ is the set of behavioral modes $b_i \in \mathbf{B}_i$ such that b_i corresponds to some state $\widetilde{C}_i$ in s_j. Given a diagnosis $\Delta(\sigma)$, $\Delta^F(\sigma)$ is the set of fault modes in $\Delta(\sigma)$.*

A candidate diagnosis is thus an assignment of a mode to each one of the components and we can show that the set of candidate diagnoses according to Definition 2 coincides with the diagnoses that would be computed by abductive diagnosis [4]. In analogy with the abductive approach, we can also define a notion of *minimal diagnosis*, i.e., a diagnosis that contains a non-redundant set of fault modes [4].

Let us consider now the case of multiple snapshots. In this case a path σ can be partitioned into a sequence $\sigma_1, \ldots, \sigma_k$ of sub-paths, each one corresponding to a single snapshot. We thus have that a diagnosis $\Delta(\sigma)$ is a sequence of single-snapshot diagnoses $\Delta_1(\sigma), \ldots \Delta_k(\sigma)$, where each $\Delta_i(\sigma)$ is obtained from σ_i according to Definition 2.

6 Specifying and testing diagnosability

In this section we show that PEPA is a suitable framework for testing the diagnosability of a system. Intuitively, studying the diagnosability of a system corresponds to studying if it can be diagnosed given the available sensors, the modeling abstractions, a number of snapshots of sensor readings. This is an interesting property and the ability to verify it would be very important in different tasks. For example, during the design of a system it can be used to decide the number, position and type of sensors; during diagnostic software development it can be used to evaluate alternative modeling assumptions.

Since a general discussion on diagnosability would be too long, we consider here some simple cases giving the flavor of the adequacy of PEPA for such an analysis. In particular, we assume that we are only interested in finding an optimal sensor placement for diagnosability [20].

Definition 3 *We say that a system is diagnosable with a given set $\mathcal{S}e$ of sensors if and only if (i) for any relevant combination of sensors readings there is only one minimal candidate diagnosis and (ii) all faults of the system correspond to a candidate diagnosis for some sensor reading.*

By *relevant* we mean that we consider only the combinations that can occur in reality[4]. In terms of PEPA this property can be tested by studying the equations

$$Diag_i \stackrel{def}{=} Obs_i \underset{S\cup\{end\}}{\bowtie} SD$$

for different sets of sensors $\mathcal{S}e$ and the corresponding sets S of observable actions, and for all relevant combinations of observations Obs_i for the selected sensors. This study has to be repeated for different set of sensors until we find the set that makes the system diagnosable. Notice that in the equations $Diag_i$, the PEPA model SD of the system under investigation is left unchanged. Varying the set of sensors corresponds to varying the observation and the sets S and H.

A system is diagnosable if for all Obs_i and the corresponding LTS_{Diag_i}, we have that for each LTS_{Diag_i} the paths have the following property:

Definition 4 *Given a set $S = \{\sigma_1, \ldots, \sigma_n\}$ of paths, we say that the paths in S are* d-equivalent *iff $\exists j$ such that $D^F(\sigma_j) = \bigcap_{i=1..n} D^F(\sigma_i)$.*

that is, d-equivalent paths correspond to the same minimal candidate diagnosis; the computation of d-equivalence can be made automatic using the PEPA Workbench.

Property 1 *A system SD is* diagnosable *with respect to a set $\mathcal{S}e$ of sensors with corresponding observables actions S if (i) for all relevant observations Obs_i we have that all consistent paths in the LTS of $Obs_i \underset{S\cup\{end\}}{\bowtie} SD$ are d-equivalent and (ii) all faults correspond to at least one path for some Obs_i.*

Let us consider again the example of Fig.1. We have three potential sensors $flow_P, level_{TA}, level_{CO}$ and we are interested in finding which minimal subset(s) of them is sufficient for having a diagnosable system. We first start by studying the cases where we use only one sensor at a time. For example, if we consider sensor $level_{TA}$, we can have only three relevant observations

$Obs_1 \stackrel{def}{=} nrm_{ta}.end.End \qquad Obs_2 \stackrel{def}{=} low_{ta}.end.End$
$Obs_3 \stackrel{def}{=} zro_{ta}.end.End$

Thus we have to study three equations of the form:

[4] In some cases the values of sensors readings may be related or constrained, hence not all the combinations are significant.

$$Diag_i \stackrel{def}{=} Obs_i \underset{S\cup\{end\}}{\bowtie} SD_1 \quad S = \{nrm_{ta}, low_{ta}, zro_{ta}\}$$

It turns out that for Obs_2 the system is not diagnosable since LTS_{Diag_2} contains two non d-equivalent paths corresponding to the two minimal diagnosis $D_1 = \{P^{(1)}\ \mathsf{leaking}\}$ and $D_2 = \{PI^{(1)}\ \mathsf{leaking}\}$.

On the other hand, it can be shown that when using the two sensors $level_{TA}$ and $level_{CO}$ the system is diagnosable. In fact, for any combination of values of these two sensors all paths in the corresponding LTS_{Diag_i} are d-equivalent. Item *(ii)* of property 1, i.e. the fact that each fault corresponds to at least one path, can be easily verified as well.

More generally, since *Obs* can be any algebraic expression, we can test other dimensions concerning diagnosability. For example, one could check if taking into account the relative ordering of snapshots is relevant for diagnosis (i.e., if a state-based approach is sufficient [22]). This can be done by checking if the two following expressions lead to the same set of diagnoses for all relevant snapshots Obs_i and Obs_j

$$Diag_{ij} \stackrel{def}{=} Obs_i.Obs_j \underset{S\cup\{end\}}{\bowtie} SD$$
$$Diag_{ji} \stackrel{def}{=} Obs_j.Obs_i \underset{S\cup\{end\}}{\bowtie} SD$$

7 Conclusions

We have shown that process algebras (and PEPA in particular) are suitable techniques for model-based diagnosis and for studying diagnostic properties of physical systems. What we presented can be regarded as a preliminary work since we have not exploited all the potentialities of the formalism yet. In particular, we will investigate the exploitation of deterministic and stochastic time algebraic extensions to deal with time in physical systems and in the diagnostic process. Moreover, we will investigate how more complex properties concerning diagnosis and diagnosability can be defined and verified within our framework, taking advantage of the possibility of using any equation for representing the observations.

To the best of our knowledge, similar approaches have been considered in [18, 19, 16]. In [18] the behavior of system components is modeled with automata and the system description is obtained through automata composition. This kind of description is indeed similar to the LTS described in a previous section; a major difference is that the authors of [18] describe components as discrete event systems, whereas we use structure and function models. In [18] diagnosis and diagnosability are defined starting from the notion of a *diagnoser*; this is an automaton that takes an observation as input while its final state represents the diagnosis. The diagnoser introduces a further step in the diagnostic process, since it has to be compiled; moreover it depends on the set of observables, and therefore has to be rebuilt when this set changes. The complexity of the diagnoser depends on the assumptions on the observable events. [16] exploits the diagnoser approach too, though starting from a system description based on temporal communicating automata.

In our case the system description takes the role of the diagnoser as well; once it is synchronized with an observation it is able to perform diagnosis. Notice also that this is done without making assumptions on which parameters can be observed, or for example on whether the order of observations is known or not. The choice about that is left to the observer, who is represented by an algebraic expression and is simply composed with the system itself. The same holds for definition and verification of system diagnosability.

What comes out is that PEPA allows an intensional and truly compositional description of the system to be diagnosed; separating the system itself from the observer leads to flexible notions of diagnosis and diagnosability; last but not least, it comes with a suite of tools for testing system properties which can be extended for our diagnosis purposes.

Acknowledgements The authors thanks all the members of the PEPA group at the University of Edinburgh for their support.

REFERENCES

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, John Wiley, 1995.
[2] V. Brusoni, L. Console, P. Terenziani, and D. Theseider Dupré, 'A spectrum of definitions for temporal model-based diagnosis', *Artificial Intelligence*, **102**(1), 39–79, (1998).
[3] E.M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, The MIT Press, 1999.
[4] L. Console and P. Torasso, 'A spectrum of logical definitions of model-based diagnosis', *Comput. Intell.*, **7**(3), 133–141, (1991).
[5] J. de Kleer, A. Mackworth, and R. Reiter, 'Characterizing diagnoses and systems', *Artificial Intelligence*, **56**(2–3), 197–222, (1992).
[6] S. Gilmore and J. Hillston, 'The PEPA workbench: A tool to support a Process Algebra based approach to performance modelling', in *Proc. 7th Int. Conf. on Mod. Techniques and Tools for Computer Perf. Eval.*, volume 794 of *LNCS*, (1994).
[7] *Readings in Model-Based Diagnosis*, eds., W. Hamscher, L. Console, and J. de Kleer, Morgan Kaufmann, 1992.
[8] J. Hillston, *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.
[9] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985.
[10] M. Malhotra and K. Trivedi, 'Dependability modeling using Petri nets', *IEEE Trans. on Reliability*, **44**, 428–440, (1995).
[11] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.
[12] R. Patton, P. Frank, and R. Clark, *Fault Diagnosis in Dynamic Systems, theory and applications*, Prentice Hall, 1989.
[13] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[14] D. Poole, 'Normality and faults in logic-based diagnosis', in *Proc. 11th IJCAI*, pp. 1304–1310, Detroit, (1989).
[15] R. Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **32**(1), 57–96, (1987).
[16] L Rozé, *Supervision de Réseaux de Télécommunication: une approche à base de modèles*, Ph.D. dissertation, Univ. de Rennes I, 1997.
[17] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, 'A discrete event systems approach to failure diagnosis', in *Proc. 5th Int. Work. on Principles of Diagnosis*, pp. 269–277, (1994).
[18] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, 'Diagnosability of discrete event systems', *IEEE Trans. on Aut. Contr.*, **40**(9), 1555–1575, (1995).
[19] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, 'Failure diagnosis using discrete event models', *IEEE Tr. on Contr. Sys. Tech.*, **4**(2), 105–124, (1996).
[20] E. Scarl, 'Sensor placement for diagnosability', *Annals of Math. and AI*, **11**(1-4), 493–510, (1994).
[21] P. Struss, 'What's in SD? Towards a theory of modeling for diagnosis', In Hamscher et al. [7], 419–449.
[22] P. Struss, 'Fundamentals of model-based diagnosis of dynamic systems', in *Proc. IJCAI 97*, pp. 480–485, (1997).