Diagnosis and diagnosability analysis using Process Algebras

Luca Console, Claudia Picardi, Marina Ribaudo
Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy,
email: {lconsole,picardi,marina}@di.unito.it

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.

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 (Hamscher, Console, & de Kleer 1992) or (Brusoni *et al.* 1998; Bylander *et al.* 1991; de Kleer, Mackworth, & Reiter 1992; Lucas 1998)). 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 model (Struss 1992) or sensor placement for diagnosability (Scarl 1994) or, more generally, diagnosability (Sampath *et al.* 1995).

Although logic is the formalism most frequently used in AI frameworks for diagnosis (Reiter 1987; Poole 1989; Console & Torasso 1991), the comparison with other formalisms such as FDI techniques (Patton, Frank, & Clark 1989) or discrete event systems (Sampath *et al.* 1994; Rozé 1997), is attracting increasing attention. Indeed, there is a long tradition in the use of discrete event systems for modeling complex physical systems and for analyzing their functional properties (Milner 1989; Peterson 1981), their performance (Hillston 1996; Ajmone Marsan *et al.* 1995), reliability and fault tolerance (Malhotra & Trivedi 1995). However, only in a few cases, these approaches have been used in the context of diagnosis and diagnosability (see e.g., (Sampath *et al.* 1995; Rozé 1997)).

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 (Hillston 1996), the one we selected) 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 analysis).

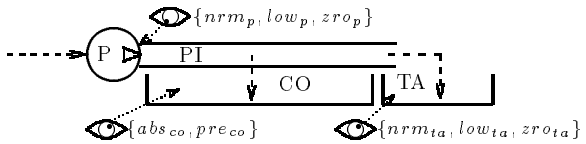

Figure 1: Abstract representation of the example

The paper is organized as follows. We first describe a simple device that will be used as a running example throughout the paper. We then briefly introduce PEPA and we show how this algebra supports the modeling of complex physical systems and the characterization of diagnosis. In order to support this claim we show a portion of the model of the Common Rail fuel delivery system, which was used as one of the guiding applications in the VMBD project (Cascio *et al.* 1999). Finally, we briefly sketch how process algebras can be applied to the analysis of properties of a physical system; in particular, we give the flavor of how they can be used for investigating diagnosability (sensor placement for diagnosability). We conclude the paper with comparisons to related work.

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 can be added to the device (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 the fourth section of the paper. For the sake of simplicity we consider a model in which dynamics is abstracted, i.e., we consider a stateless abstraction in which the level of water in the tank depends only on the input to the tank. We then use the Common Rail example to show how dynamic behavior can be modeled. Throughout the paper we regard time as a sequence of states (events), without any metric (see the discussion in (Brusoni *et al.* 1998)).

PEPA

In this section we briefly introduce the syntax of Performance Evaluation Process Algebra (PEPA) (Hillston 1996), an algebraic description technique based on classical process algebras (Hoare 1985; Milner 1989) and enhanced with timing information. This extension results in models which may be used to calculate performance measures as well as to 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[1]. The grammar of the language is shown below, the names and the meanings of the operators are informally presented referring to our running example.

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

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 between the two components, 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 in the grammar above, H represents the set of actions that are hidden; for example, in $P/\{nrm_0\}$ the action nrm_0 is hidden.

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:

[1] All the details can be found in (Hillston 1996) or at the PEPA Web page www.dcs.ed.ac.uk/pepa.

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 equations describing the behavior of a generic component type T have the following structure: a first equation defines the modes of behavior of T and is written as a nondeterministic choice among its possible modes. Then for each mode there is a set of equations. In particular, the actions of PEPA are used to express conditions on input, output (and state) variables.

Let us consider the example of Fig.1; the equations describing the behavior of the component types in this simple device are reported in Fig.2. The equations P, PI, TA and CO define respectively, the behavior of the generic component types pump, pipe, tank and collector. 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 a pipe. The additional identifier End allows the component to evolve into a final (absorbing) state whose meaning will be clear in the next section when we shall characterize diagnoses.

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)[2].

The equations describing a component are recursive: in this way we can describe its behavior across time. As we noticed, 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.

[2]The pedices are used to distinguish the two outputs of the pipe.

Components behavior

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

Structure of the device

component instances

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

connections

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

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

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

The example of Fig. 1 is composed of a pump, a pipe, a tank, a collector, which are declared as instances of the corresponding component types. We use indeces to denote instances; $P^{(1)} : P$ in Fig. 1 means that $P^{(1)}$ is an instance of the component type P.

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 $TA^{(1)}$ and $CO^{(1)}$, are now invisible.

The semantics of each (untimed) PEPA term is given via a *Labeled Transition System* LTS $= (\mathcal{C}, \mathcal{A}, \stackrel{\alpha}{\longrightarrow})$

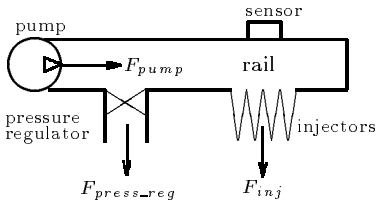

Figure 3: The Common Rail System

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. Qualitative properties, such as the absence of deadlocks or the reachability of a given state, can be studied by investigating it. Model-checking techniques (Clarke, Grumberg, & Peled 1999) can also be used to verify system properties expressed using logical formulas.

The LTS of PEPA models can be automatically computed using the PEPA Workbench (Gilmore & Hillston 1994), a prototype tool which supports modeling and analysis with PEPA. In the case of SD_1 (Fig. 2) it consists of 337 states and 842 transitions and describes all its possible behaviors. In the following we will use LTS_{SD_1} to denote this transition system.

Modeling a real system

We end this section by briefly presenting a portion of the model of a real system. More specifically, we consider the *Common Rail fuel delivery system*, which has been used as one of the guiding applications in the VMBD project (see (Cascio *et al.* 1999) for a description of the system, which is schematized in Fig. 3).

We consider one of the components, the *rail*, an accumulator in which fuel is maintained at a very high pressure. The fuel is delivered to the rail by a pump and the pressure in the rail is controlled by an Electronic Control Unit (ECU). Given the current value of the pressure in the rail (P_{rail}), the driving conditions (e.g., the position of the pedals, the engine speed, etc.) and an internal strategy, the ECU computes the desired value for the pressure in the rail and controls a pressure regulating valve, opening or closing it (i.e., deciding the flow $F_{press-reg}$ through the valve) in order to achieve such a desired pressure. The ECU controls also the amount of fuel delivered to the injectors in the cylinders.

In a model of the correct behavior of the rail we are interested in describing how the pressure of the fuel in the rail varies according to (i) the input it receives from the pump and (ii) the output through the pressure regulating valve and the injectors. The pressure can be modeled as a state variable whose change is influenced by the balance between input and output flows (and which in turn influences these flows directly or via the ECU). Thus, this example allows us to show how a dynamic system (with feedback) can be modeled using PEPA.

The model designed in (Cascio *et al.* 1999) is based on qualitative deviations (see also (Malik & Struss 1996)). Very roughly, this means that for each variable X we consider the sign of the deviation of X with respect to the normal (expected) value it should assume; this will be denoted as $[\Delta X]$. For example, the equation describing the correct behavior of the rail is

$$\delta P_{rail} = [\Delta F_{pump}] - [\Delta F_{Press-reg}] - [\Delta F_{inj}]$$

where:

- signs can assume the three values zero, plus and minus;
- "$-$" is the difference in the sign algebra;
- δP_{rail} denotes the sign of the deviation of the derivative; for example, if $\delta(P_{rail}) =$ minus, then the derivative is less than it should be.

Thus, the equation above allows us to determine δP_{rail}, given the deviations of the flows and then to determine how the state variable $[\Delta P_{rail}]$ changes from one state to its successor (recall that we are modeling time as a sequence of states).

For example, if $[\Delta F_{pump}] =$ plus, i.e., the flow from the pump is more than it should be, $[\Delta F_{Press-reg}] =$ zero and $[\Delta F_{inj}] =$ zero, i.e., these flows are as expected, then δP_{rail} is positive, i.e., the pressure in the rail grows more than it should.

The PEPA model for the correct behavior of the rail is reported in Fig.4.

Equation (i) specifies that the behavior of the rail is different given the current value of its state variable ΔP_{rail}: $Raok_2$ corresponds to the case where the pressure is as expected ($zero_{\Delta P_{rail}}$), $Raok_3$ and $Raok_4$ to the cases where it is less than expected or more than expected, respectively. In equation (ii) we consider the flow from the pump to the rail; for example in case this flow is less than expected ($minus_{\Delta F_{pump}}$), the system evolves into $Raok_5$ and similarly for the other cases (for the sake of brevity we reported only a subset of the equations, the other being similar).

In $Raok_5$ we consider the output through the pressure regulating valve; when this output is as expected ($zero_{\Delta F_{press-reg}}$) or more than expected ($plus_{\Delta F_{press-reg}}$), the system moves into $Raok_8$, otherwise into $Raok_9$. Similarly, starting from $Raok_8$ it moves into $Raok_{10}$ when the flow to the injectors is as expected or more than expected, and it moves into $Raok_{11}$ when it is less than expected.

In summary, the system moves into $Raok_{10}$ in case $[\Delta F_{pump}] =$ minus, $[\Delta F_{press-reg}] =$ zero or plus, $[\Delta F_{inj}] =$ zero or plus. This means that $\delta P_{rail} =$minus and, since we are considering the case where $[\Delta P_{rail}] =$ zero, ($zero_{\Delta P_{rail}}$ in equation (i)), we have that in $Raok_{10}$ $[\Delta P_{rail}]$ must change to minus, because the sign of the deviation of its derivative is negative. From $Raok_{10}$ the system moves back to $Rail_1$ producing $minus_{\Delta P_{rail}}$ (and $minus_{\Delta P_{sensor}}$ to the pressure sensor). In the case of $Raok_{11}$, on the other hand, we have

$$
\begin{array}{lll}
 & Rail_1 & \stackrel{def}{=} Raok_1 + Ralk_1 + End \\
(i) & Raok_1 & \stackrel{def}{=} zero_{\Delta P_{rail}}.Raok_2 + minus_{\Delta P_{rail}}.Raok_3 + plus_{\Delta P_{rail}}.Raok_4 \\
(ii) & Raok_2 & \stackrel{def}{=} minus_{\Delta F_{pump}}.Raok_5 + zero_{\Delta F_{pump}}.Raok_6 + plus_{\Delta F_{pump}}.Raok_7 \\
 & Raok_3 & \stackrel{def}{=} \ldots \\
 & Raok_4 & \stackrel{def}{=} \ldots \\
(iii) & Raok_5 & \stackrel{def}{=} zero_{\Delta F_{press-reg}}.Raok_8 + plus_{\Delta F_{press-reg}}.Raok_8 + minus_{\Delta F_{press-reg}}.Raok_9 \\
 & Raok_6 & \stackrel{def}{=} \ldots \\
 & Raok_7 & \stackrel{def}{=} \ldots \\
(iv) & Raok_8 & \stackrel{def}{=} zero_{\Delta F_{inj}}.Raok_{10} + plus_{\Delta F_{inj}}.Raok_{10} + minus_{\Delta F_{inj}}.Raok_{11} \\
 & Raok_9 & \stackrel{def}{=} \ldots \\
(v) & Raok_{10} & \stackrel{def}{=} minus_{\Delta P_{rail}}.minus_{\Delta P_{sensor}}.Rail_1 \\
(vi) & Raok_{11} & \stackrel{def}{=} minus_{\Delta P_{rail}}.minus_{\Delta P_{sensor}}.Rail_1 + plus_{\Delta P_{rail}}.plus_{\Delta P_{sensor}}.Rail_1 + \\
 & & \qquad zero_{\Delta P_{rail}}.zero_{\Delta P_{sensor}}.Rail_1 \\
 & \ldots & \ldots \\
(x) & Rail_2 & \stackrel{def}{=} y.y.Rail_2 \\
(xi) & Rail & \stackrel{def}{=} Rail_1 \underset{L_3}{\bowtie} Rail_2 \text{ where } L_3 = \{zero_{\Delta P_{rail}}, plus_{\Delta P_{rail}}, minus_{\Delta P_{rail}}\} \\
 & \ldots & \ldots \\
(xii) & SD_2 & \stackrel{def}{=} (Pump \parallel PressReg \parallel Injector) \underset{L_4}{\bowtie} Rail \underset{L_5}{\bowtie} Sensor \\
 & & \text{where } L_4 = \{zero_{\Delta F_{pump}}, plus_{\Delta F_{pump}}, minus_{\Delta F_{pump}}, \\
 & & \qquad zero_{\Delta F_{press-reg}}, plus_{\Delta F_{press-reg}}, minus_{\Delta F_{press-reg}}, \\
 & & \qquad zero_{\Delta F_{inj}}, plus_{\Delta F_{inj}}, minus_{\Delta F_{inj}}\} \\
 & & and\ L_5 = \{zero_{\Delta P_{sensor}}, plus_{\Delta P_{sensor}}, minus_{\Delta P_{sensor}}\}
\end{array}
$$

Figure 4: Equations describing the correct behavior of the Common Rail.

that $[\Delta F_{pump}] = \mathsf{minus}$, $[\Delta F_{press-reg}] = \mathsf{zero}$ or plus and $[\Delta F_{inj}] = \mathsf{minus}$. Due to the ambiguity in the sign algebra, we have that δP_{rail} can assume any value in $\{\mathsf{minus}, \mathsf{zero}, \mathsf{plus}\}$ and then $[\Delta P_{rail}]$ may as well change accordingly (equation (vi)).

In order to model the dynamic behavior of the rail, equation (x) introduces a dummy component $Rail_2$ which is used in (xi) to synchronize the output from the rail to the input to the rail itself. The notation $y.y.Rail_2$ in (x) is used to represent that $Rail_2$ simply reflects its input (whatever the value of the input is) to its output; y can be regarded as a sort of variable ranging over the events $zero_{\Delta P_{rail}}, plus_{\Delta P_{rail}}, minus_{\Delta P_{rail}}$.

Finally, equation (xii) gives an idea of the structure of the whole device, introducing all the components and their interactions.

A characterization of diagnosis

In this section we discuss how diagnoses can be characterized in our framework. 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. $Obs_2 \stackrel{def}{=} nrm_p.Obs'_2 + low_p.Obs'_2$; $Obs'_2 \stackrel{def}{=} abs_{co}.nrm_{ta}.end.End$ 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.
4. $Obs_4 \stackrel{def}{=} nrm_p.abs_{co}.nrm_{ta}.nrm_p.pre_{co}.low_{ta}.end.End + nrm_p.pre_{co}.low_{ta}.nrm_p.abs_{co}.nrm_{ta}.end.End$ corresponds to a case where the ordering between the two snapshots is not known.

Given a system description SD and an observation Obs,

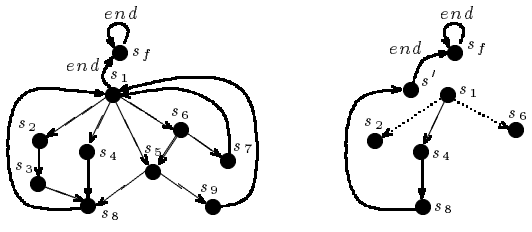

Figure 5: LTS_{SD} (left) and LTS_{Diag} (right)

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.

A pictorial representation of an LTS_{SD} and the corresponding LTS_{Diag} is shown in Fig.5. Only the paths leading the system to the state enabling action end (see solid arcs in LTS_{Diag}) correspond to evolutions which are *consistent* with the observations and are those of interest for our diagnosis purpose. Other evolutions, which were possible in SD before synchronization (see dotted arcs in LTS_{Diag}, Fig.5) are not allowed anymore since they correspond to behaviors which are not completely consistent with 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 diagnostic process. 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 $D(\sigma)$ 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 $D(\sigma)$, $D^F(\sigma)$ is the set of fault modes in $D(\sigma)$.*

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 (Console & Torasso 1991). 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 (Console & Torasso 1991).

Let us consider now the case of multiple snapshots. In this case a path σ can be partitioned into a sequence $\langle \sigma_1, \ldots, \sigma_k \rangle$ of sub-paths, each one corresponding to a single snapshot. We thus have that a diagnosis $D(\sigma)$ is a sequence $\langle D_1(\sigma), \ldots D_k(\sigma) \rangle$ of single-snapshot diagnoses, where each $D_i(\sigma)$ is obtained from σ_i according to Definition 2.

Let us consider again the example in Fig. 1. Given the observation $Obs_1 \stackrel{def}{=} nrm_p.pre_{co}.low_{ta}.end.End$, we have the only minimal diagnosis $D_1 = \{P^{(1)}\ \mathsf{ok}, PI^{(1)}\ \mathsf{leaking}\}$. On the other hand, given the observation $Obs_3 \stackrel{def}{=} nrm_p.abs_{co}.nrm_{ta}.nrm_p.pre_{co}.low_{ta}.end.End$, we have the diagnosis $\langle D_0, D_1 \rangle$, where $D_0 = \{P^{(1)}\ \mathsf{ok}, PI^{(1)}\ \mathsf{ok}\}$ and D_1 is as above.

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 (Scarl 1994).

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 belong to a candidate diagnosis for some sensor reading.*

By *relevant* we mean that we consider only the combinations that can occur in reality[3]. 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 sets 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, the paths are d-equivalent if they 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 for all relevant observations Obs_i we have that (i) all consistent paths in the state space of $Obs_i \underset{S \cup \{end\}}{\bowtie} SD$ are d-equivalent and (ii) all faults that must be diagnosed correspond to at least one path.*

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

$Obs_1 \stackrel{def}{=} nrm_{ta}.end.End$

$Obs_2 \stackrel{def}{=} low_{ta}.end.End$

$Obs_3 \stackrel{def}{=} zro_{ta}.end.End$

Thus we have to study three equations

$Diag_i \stackrel{def}{=} Obs_i \underset{S \cup \{end\}}{\bowtie} SD_1$

$S = \{nrm_{ta}, low_{ta}, zro_{ta}\}$

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

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.

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 (Struss 1997)). 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

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

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

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 (Sampath *et al.* 1995; 1996; Rozé 1997). In (Sampath *et al.* 1995) 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 (Sampath *et al.* 1995) describe components as discrete event systems, whereas we use structure and function models. In (Sampath *et al.* 1995) 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

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

set changes. The complexity of the diagnoser depends on the assumptions on the observable events.

(Rozé 1997) exploits the diagnoser approach too, though starting from a system description through 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 This work started when M. Ribaudo was visiting the LFCS at the University of Edinburgh, supported by a grant from CNR. The authors thanks all the members of PEPA group for their support.

References

Ajmone Marsan, M.; Balbo, G.; Conte, G.; Donatelli, S.; and Franceschinis, G. 1995. *Modelling with Generalized Stochastic Petri Nets.* John Wiley.

Brusoni, V.; Console, L.; Terenziani, P.; and Theseider Dupré, D. 1998. A spectrum of definitions for temporal model-based diagnosis. *Artificial Intelligence* 102(1):39–79.

Bylander, T.; Allemang, D.; Tanner, M.; and Josephson, J. 1991. The computational complexity of abduction. *Artificial Intelligence* 49(1-3):25–60.

Cascio, F.; Console, L.; Guagliumi, M.; Osella, M.; Panati, A.; Sottano, S.; and Theseider Dupré, D. 1999. Generating on-board diagnostics of dynamic automotive systems based on qualitative deviations. *AI Communications* 12(1):33–44. Also in Proc. 10th Int. Workshop on Principles of Diagnosis, Loch Awe, Scotland.

Clarke, E.; Grumberg, O.; and Peled, D. 1999. *Model Checking.* The MIT Press.

Console, L., and Torasso, P. 1991. A spectrum of logical definitions of model-based diagnosis. *Comput. Intell.* 7(3):133–141.

de Kleer, J.; Mackworth, A.; and Reiter, R. 1992. Characterizing diagnoses and systems. *Artificial Intelligence* 56(2–3):197–222.

Gilmore, S., and Hillston, J. 1994. 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*.

Hamscher, W.; Console, L.; and de Kleer, J., eds. 1992. *Readings in Model-Based Diagnosis.* Morgan Kaufmann.

Hillston, J. 1996. *A Compositional Approach to Performance Modelling.* Cambridge University Press.

Hoare, C. 1985. *Communicating Sequential Processes.* Prentice-Hall.

Lucas, P. 1998. Analysis of notions of diagnosis. *Artificial Intelligence* 105(1-2):295–343.

Malhotra, M., and Trivedi, K. 1995. Dependability modeling using Petri nets. *IEEE Transactions on Reliability* R-44:428–440.

Malik, A., and Struss, P. 1996. Diagnosis of dynamic systems does not necessarily require simulation. In *Proc. 7th Int. Work. on Principles of Diagnosis*, 147–156.

Milner, R. 1989. *Communication and Concurrency.* Prentice Hall.

Patton, R.; Frank, P.; and Clark, R. 1989. *Fault Diagnosis in Dynamic Systems, theory and applications.* Prentice Hall.

Peterson, J. 1981. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall.

Poole, D. 1989. Normality and faults in logic-based diagnosis. In *Proc. 11th IJCAI*, 1304–1310.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–96.

Rozé, L. 1997. *Supervision de Réseaux de Télécommunication: une approche à base de modèles.* Ph.D. Dissertation, Univ. de Rennes I.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Teneketzis, D. 1994. A discrete event systems approach to failure diagnosis. In *Proc. 5th Int. Work. on Principles of Diagnosis*, 269–277.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Teneketzis, D. 1995. Diagnosability of discrete event systems. *IEEE Trans. on Aut. Contr.* 40(9):1555–1575.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Teneketzis, D. 1996. Failure diagnosis using discrete event models. *IEEE Trans. on Contr. Sys. Tech.* 4(2):105–124.

Scarl, E. 1994. Sensor placement for diagnosability. *Annals of Math. and AI* 11(1-4):493–510.

Struss, P. 1992. What's in SD? Towards a theory of modeling for diagnosis. In Hamscher, W.; Console, L.; and de Kleer, J., eds., *Readings in Model-based Diagnosis.* Morgan Kaufman. 419–449.

Struss, P. 1997. Fundamentals of model-based diagnosis of dynamic systems. In *Proc. IJCAI 97*, 480–485.