# Distributed Consistency-Based Diagnosis without Behavior

**Gianfranco Lamperti** [1], **Marina Zanella** [1]

[1] *Dipartimento di Ingegneria dell'Informazione, Brescia, 25123, Italy*
*gianfranco.lamperti@ing.unibs.it*
*marina.zanella@ing.unibs.it*

## ABSTRACT

This paper faces the task of distributed diagnosis without exploiting any component behavioral model. A diagnosis problem is specified by an observation that just states whether each system output is either correct or incorrect. The system is split into parts, and a distinct diagnoser, which is supplied with knowledge that has been compiled off-line and is capable of communicating with its neighbors, is assigned to each of them. A family of methods is proposed to compute local and global minimal diagnoses that are consistent with both the observation and the system description, the latter being a kind of control flow structure.

## 1 INTRODUCTION

A consistency-based diagnosis (Reiter, 1987) is a (minimal) set of system components such that assuming that all the other components are behaving normally is consistent with the system description and the observation. This definition, as well other equivalent ones, requires the model of the normal behavior of every component to be available, along with the structure and observation of the considered system, the latter encompassing all input and output values. Cannot the task be simplified just considering the correctness of the outputs, without taking into account their specific values? What are the relationships between a *valued* consistency-based diagnosis and an *unvalued* one? What is the impact of an unvalued approach on distributed diagnosis?

Consistency-based diagnosis exploits the models of the normal behavior of system components in order to predict the system correct outputs based on the given (implicitly correct) inputs so as to find out which are the incorrect outputs, if any, of the real system to be diagnosed. However, this is not the only way to ascertain the (in)correctness of the output values of a system. The same result can be obtained by proving whether preconditions and postconditions hold. Preconditions constrain the correct input values, and postconditions link the correct output values to the (correct) input values. If input values are implicitly correct, only postconditions have to be considered. Moreover, there exist systems for which the (in)correctness of output values can be stated by an oracle. For instance, in a choreography of business processes, the users are capable of stating whether an output is either correct or incorrect. Therefore, if either an oracle or the preconditions and postconditions of the system are available, the models of the normal behavior of system components are not needed to find out which outputs are incorrect. Cannot we do without them also to compute diagnoses, both in a centralized and in a distributed way?

This paper raises the above questions and reports some preliminary thoughts on the topic. The setting for distributed diagnosis is similar to that dealt with in (Armant *et al.*, 2008), where *peers*, that is, reasoning agents, each inherent to a subsystem of a distributed system, only know their neighbors. However, the quoted work, differently from ours, exploits the normal behavior of system components.

The present work has been inspired by (Borrego *et al.*, 2009), from which it borrows the notion of a *signature matrix* and (in some respect) a diagnostic method, while both providing a theoretical foundation and extending it. In (Borrego *et al.*, 2009) a single method to compute minimal local diagnoses is proposed, this paper instead proposes a family of methods to achieve both local and global minimal diagnoses. The next section describes the adopted modeling primitives, while Section 3 deals with the generation of compiled knowledge, typically the aforementioned signature matrices. The diagnosis method, which circumscribes its attention to diagnoses without masking phenomena, is then illustrated both in a monolithic scenario in Section 4 and in a distributed scenario in Section 5. Some final remarks are recorded in the concluding section.

## 2 SYSTEM MODELING

A composed system consists of components. Each component has its own behavior, which may be static or dynamic, possibly a cyclic one; however, no knowledge of the (normal and/or abnormal) behavior is available to the diagnosis task. The only assumption inherent to the behavior is that any execution of a compo-

nent, which is called an *activity*, takes a finite time. Distinct executions of the same component can be represented as distinct activities. The diagnosis task is carried out once all the activities relevant to the considered system input value(s) have finished. A diagnosis problem inherent to a given system is specified by the system observation, which states whether inputs and outputs are either correct ($OK$) or incorrect ($KO$). Owing to the lack of any behavioral models, the diagnosis output can only pinpoint the possibly faulty activities, not the specific faults affecting components. The only model that is assumed to be available describes the control flow between activities. Very few modeling primitives are needed: the activity, the flow, a structure for branching a flow into several mutually exclusive flows, a structure for merging several mutually exclusive flows into one flow, a structure for forking a flow into several parallel flows, and a structure for joining several parallel flows into one flow. A model resembles a UML (Fowler, 2004) activity diagram, it has the same meaning and, actually, the same graphics could be adopted. However, a model is an activity diagram without any guards since the diagnosis task cannot find out the value of any condition at a branching point of the execution owing to the lack of the component behavioral models. Moreover, a model does not include any cycle as, for the same reason, the diagnostic task cannot devise the values of the cycle (initial/final) conditions, the number of iterations or which iteration may have behaved abnormally. Thus, if in the portion of the real world we are modeling there is a loop, just its body has to be modeled. In case the loop execution may amount to no iteration, the model has to include two mutually exclusive paths: the former containing the sequence of activities representing such a body and the latter containing just a flow (with no activity). An activity has at least an input flow and produces an output flow. All the input flows of an activity are implicitly joined. Depending on the modeler's purpose, such a model can feature different levels of generality: it may either represent (in a concise and acyclic way) all the possible executions of a system or just some of them. Such executions are triggered by system inputs.

Some interconnected activities form a *system*. The notion of a system requires that no activity is isolated: an input flow (at least) of an activity comes from another activity and/or its output flow is directed to another activity. Thus a system model amounts to a connected DAG (Directed Acyclic Graph) of activities which is endowed with inputs and outputs, coming from and directed to outside the system. Each system input is an input flow of an activity, and each system output is the output flow of an activity.

The modeling language briefly described above, owing to its simplicity and generality, can be exploited to model at a very high abstraction level several (static and dynamic) systems, such as combinatorial networks, software programs, industrial workflows, biological processes, etc.

**Example 1** Figure 1 displays system $T$, which will be considered throughout the paper. Square are activities, arrows are flows and the diamond is a fork. This model is compatible with the well known device introduced in (Davis, 1984) and analyzed in (Reiter, 1987; de Kleer and Williams, 1987), where $M1$, $M2$ and $M3$ are multipliers and $A1$ and $A2$ are adders. However, the approach presented here does not care whether a component is either a multiplier or an adder or something else: the same reasoning is performed for whichever physical device that shares this model, independently of the specific behavior of the components at hand. In particular, in the quoted references this is the structural model, where each arrow denotes a data transmission, while, according to the approach described here, an arrow is a control flow. Thus, out of determinism, each activity has just one output flow, while a component in (Reiter, 1987; de Kleer and Williams, 1987) may have several data outputs. Actually, an activity handles data and a control flow may imply also a data flow, however, since the approach does not take into account any data values, we neither model the data exchange between activities nor the production of specific system outputs: each modeled system output may cumulatively represent several data outputs. Moreover, the quoted references assume that $A1$ and $A2$, for instance, are distinct components while, according to the current approach, they could be distinct executions (activities) of the same component. However, with a slight abuse of terminology, we can interchange the use of the word *component* and that of the word *activity*.

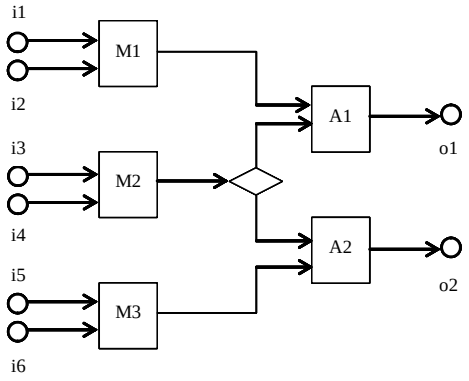


Figure 1: System $T$

In a distributed perspective, the set of the system activities can be partitioned into several parts, where each part is assigned to a *process*. Each process has its own inputs and outputs, coming from and directed to outside the process. A process input is either a system input or a flow coming from another process. Analogously, a process output is either an output of the system or a flow directed to another process.

In (Borrego *et al.*, 2009), the (larger) model of a business-to-business collaboration, including also the primitives that are not necessary to model system $T$, can be found. In such a context it is straightforward to map a process onto a business process and an activity onto an either automatic or manual operation. This is quite proper (and intuitive) to model web services.

## 3 KNOWLEDGE COMPILATION

The method proposed in (Borrego *et al.*, 2009) consists of both an off-line phase (knowledge compilation) and

an on-line phase (diagnostic problem solving). Here we adopt the same scheme. The off-line phase takes as input the system model, including its partition into processes, if any, and transforms each process into a set of clusters. If no partition is provided, only one process is considered, this being the system itself. In fact, a system is a process (while a process is not necessarily a system since it may be disconnected).

A *cluster* is a connected (sub)graph of the system DAG, representing a single execution path within the considered process, where such a path is triggered by a process input at least and ends by producing a process output, without including any mutually exclusive activities. The process inputs that trigger the cluster execution have to be the only input flows of the activities they feed. There may exist further process inputs and outputs relevant to a cluster, besides the inputs that trigger the execution path and the output produced at the end of the execution path. Since a process may consist of several disconnected graphs and each graph may include several mutually exclusive paths, there may be several clusters belonging to a single process.

A cluster can be concisely represented as a set of internal subsets, each associated with its input(s) and output. An *internal subset* is inherent to a single distinct output of the cluster at hand, and includes all and only the activities (belonging to the cluster) on which such an output depends. Thus a cluster includes as many internal subsets as the number of its outputs. Moreover, the output relevant to an internal subset depends on an input of the cluster at least.

**Example 2** System $T$ is connected (in compliance with the notion of a system) and does not include any mutually exclusive paths, therefore it corresponds to just one cluster, represented in Table 1, where each row is an internal subset.

Table 1: The only cluster of system $T$

| *Input(s)* | *Activities* | *Output* |
|---|---|---|
| $i1$, $i2$, $i3$, $i4$ | $M1$, $M2$, $A1$ | $o1$ |
| $i3$, $i4$, $i5$, $i6$ | $M2$, $M3$, $A2$ | $o2$ |

The knowledge inherent to each cluster can be compiled into a *signature matrix* (Borrego *et al.*, 2009), where each row is an internal subset and each column is an activity. A cell of the matrix is checked if the row internal subset includes the column activity. Such a matrix does not include any information about inputs if they are assumed to be correct.

**Example 3** The compilation of the knowledge about $T$ leads to the matrix in Table 2, where the output relevant to each row is the header of the row itself, and the activity relevant to each column is the header of such a column. The six cluster inputs are not represented in the matrix since they are assumed to be all $OK$.

Table 2: Matrix corresponding to the cluster in Table 1

| | $M1$ | $M2$ | $M3$ | $A1$ | $A2$ |
|---|---|---|---|---|---|
| $o1$ | √ | √ | | √ | |
| $o2$ | | √ | √ | | √ |

## 4 DIAGNOSIS

In consistency-based diagnosis a conflict is a set of components such that assuming that all of them are working normally is inconsistent with the conjunction of the system description and observation. A conflict is minimal if no proper subset of its is a conflict. In unvalued consistency-based diagnosis, in case an output is incorrect, it is impossible that all the components such an output depends on, that is, those belonging to its internal subset, behave normally. Let us call such a set a *structural conflict* since it can quite efficiently be drawn from the system model. All other (minimal) conflicts are called *non-structural* and, as it will be shown later, they exist only if there is some correct system output.

**Proposition.** A structural conflict either equals a minimal conflict of valued consistency-based diagnosis or is a superset of its.

This proposition can be justified by considering that a structural conflict is a conflict of consistency-based diagnosis if an implicit model of normal behavior is assigned to every component, according to which the value of its (only) output is $OK$ if all its inputs are $OK$, while the output is $KO$ if an input at least is $KO$. If a system output is $KO$, either the component producing such an output is faulty (having all $OK$ inputs) or any of its inputs $i$ is $KO$. This in turn means that either the component generating $i$ is faulty or any of its inputs is $KO$, and so on, till the system inputs are reached, which are assumed to be $OK$. Thus, if a system output is $KO$, each component in its internal subset may be responsible for it, and, as such, it belongs to a conflict, the so-called structural conflict. In valued consistency-based diagnosis, instead, some components in the internal subset may be exonerated. For instance, assume that the incorrect output value of a system is 0, this being the output $o$ of an OR gate, and that an input of this gate, $i$, is generated by a component $c$. If the correct value of $i$, predicted given the system input values, is 0, $c$ does not belong to any minimal conflict of valued consistency-based diagnosis since, if it were the only faulty component of the conflict, then its output would be 1, which is inconsistent with the observation (as it would force $o$ to 1 while its observed value is 0). Therefore, a structural conflict is a conflict also for valued consistency-based diagnosis but it is not a minimal conflict if some components can be exonerated based on the predicted values.

Roughly speaking, cluster level diagnoses can be computed as the hitting sets of the structural conflicts relevant to the incorrect cluster outputs. The method proposed in (Borrego *et al.*, 2009) first considers as normally working all the activities belonging to internal subsets relevant to correct outputs (this means eliminating the matrix columns representing such activities and the row columns representing such outputs), then computes the minimal hitting sets of the remaining internal subsets (they are actually structural conflicts from which the activities that are assumed to work correctly have been removed). In this section we show that this corresponds to computing diagnoses

without masking phenomena, and then throughout the paper we will consider just this kind of diagnoses.

**Example 4** If output $o1$ of $T$ is incorrect and $o2$ is correct, which is expressed as $OBS(T) = \{KO\ o1, OK\ o2\}$, row $o1$ of the matrix relevant to $T$ is marked with $KO$, while row $o2$ is marked with $OK$ along with all the activities that affect $o2$. The marked matrix is displayed in Table 3. The set of candidate diagnoses consists of the minimal hitting sets of the unmarked activities of the $KO$ rows (here row $o1$ only), that is $candidates(T;\ KO\ o1,\ OK\ o2) = \{\{M1\}, \{A1\}\}$, where the parameters of $candidates$ represent the diagnosis problem, the first being the relevant system and the following the observation. If the observation is instead $OBS(T) = \{KO\ o1,\ KO\ o2\}$, both matrix rows are marked with $KO$ while all columns are left unmarked. Then the minimal hitting sets of the two conflicts $C1 = \{M1,\ M2,\ A1\}$ and $C2 = \{M2,\ M3,\ A2\}$ have to be computed, thus producing $candidates(T;\ KO\ o1,\ KO\ o2) = \{\{M2\}, \{M1, M3\}, \{A1, M3\}, \{M1, A2\}, \{A1, A2\}\}$.

Table 3: Annotated matrix for system $T$

| | | $M1$ | $OK$ $M2$ | $OK$ $M3$ | $A1$ | $OK$ $A2$ |
|---|---|---|---|---|---|---|
| $KO$ | $o1$ | √ | √ | | √ | |
| $OK$ | $o2$ | | √ | √ | | √ |

**Proposition.** Each minimal non-structural conflict is the union of two (disjoint) parts, $Ci/ISj$ and $ISj/Ci$, where $Ci$ is a structural conflict, inherent to an incorrect output $oi$, and $ISj$ is an internal subset, inherent to a correct output $oj$, and $Ci \cap ISj \neq \emptyset$, that is, there exists a component at least that affects both outputs $oi$ and $oj$.

**Proof (sketch).** A minimal non-structural conflict $C$ is bound to include both

- a component $cj \in ISj$, as it can be proved by contradiction. Assume that no components in $C$ affect a correct output, that is, all components in $C$ are relevant to incorrect outputs. No structural conflict $Ci$ can be a subset of $C$ since $C$ is minimal, hence for each $Ci$ having a non-empty intersection with $C$, there necessarily exists a component $ci \in Ci$ at least that is not included in $C$. Since such an excluded component $ci$ is the possible reason for output $oi$ being incorrect, all the components in $Ci \cap C$ may be working normally. As a consequence, all components in $C$ may be working normally, which means that $C$ is not a conflict;
- and a component $ci \in Ci$, as it can be proved by contradiction. Assume that $C$ does not include any component $ci$ that belongs to a structural conflict. Then, all the components in $C$ affect correct outputs only and, as such, all of them may be working normally. Hence $C$ is not a conflict.

Potentially any single component in a minimal conflict may be the only component working abnormally in the conflict itself.

Suppose that $ci \in Ci$, where $Ci$ is a structural conflict, is the only component in $C$ that works abnormally. The abnormal behavior of $ci$ causes output $oi$ to be incorrect. This is consistent with the observation iff $ci$ does not affect any correct output. As a consequence, a non-structural conflict can include a component $ci$ belonging to a structural conflict only if $ci$ does not belong to an internal subset $ISj$ inherent to a correct output. In particular, if $ISj$ is overlapping $Ci$, $ci$ cannot belong to $Ci \cap ISj$, which amounts to stating that $ci$ has to belong to $Ci/ISj$. On the other hand, $C$ is bound to include all the components $ci$ in $Ci/ISj$ since there is no selection criteria among them, all of them producing the same effects on the observed output $oi$.

Dually, suppose that $cj \in ISj$, where $ISj$ is an internal subset inherent to a correct output $oj$, is the only component in $C$ that works abnormally. The abnormal behavior of $cj$ causes output $oj$ to be incorrect unless the misbehavior of $cj$ is masked by the misbehavior of another component affecting $oj$, that is by another component belonging to $ISj$. Since all the other components in $C$ are assumed to work normally, this component does not belong to $C$. Therefore, the components of $ISj$ that do not belong to $C$ are (all and only) those that affect the incorrect output $oi$, that is, those in $Ci \cap ISj$ (where $Ci$ and $ISj$ have a non-empty intersection). On the other hand, $C$ is bound to include all the components $cj$ in $ISj/Ci$ since there is no selection criteria among them, all of them producing the same effects on the observed output $oj$.

**Definition.** A (minimal) diagnosis $d$ is *with masking phenomena* if there exists an internal subset $ISj$ (inherent to a correct output $oj$) such that $|d \cap ISj| = 2$. All the other (minimal) diagnoses are *without masking phenomena*.

A diagnosis is with masking phenomena if an output at least ($oj$ in the definition) is correct owing to a fault masking another fault. The faults have to affect distinct components belonging to the set of components on which the output depends ($ISj$ in the definition). The faulty components are required to be two owing to minimality: a diagnosis including more than two faulty components affecting a correct output would not be minimal since a subset of its (including just two faulty components belonging to $ISj$) is a diagnosis. According to the same definition, a (minimal) diagnosis $d$ is without masking phenomena if there exists no $ISj$ inherent to a correct output $oj$ such that $|d \cap ISj| = 2$. However, as explained above, a minimal diagnosis cannot include more than two components of $ISj$. Moreover, $d$ cannot include just one component of $ISj$, since assuming that a single component of $ISj$ is faulty is inconsistent with the observation that $oj$ is correct. Therefore a diagnosis $d$ is without masking phenomena if it has an empty intersection with all the internal subsets inherent to correct outputs, which means that a diagnosis is without masking phenomena if it assumes that all the components correct outputs depend on are not faulty.

**Proposition.** A minimal diagnosis with masking phenomena is the hitting set of the $ISj/Ci$ part of a non-structural conflict, of $Ci \cap ISj$, and of all the difference sets $Ch/\cup IS$, where $Ch$, $h \neq i$, is a structural conflict, and $\cup IS$ is the union of all internal subsets in-

herent to all correct outputs.

**Proof (sketch).** A (minimal) diagnosis $d$ with masking phenomena, being a diagnosis, is a (minimal) hitting set of all (structural and non-structural) conflicts. However, since $d$ is with masking phenomena, it has necessarily to hit a non-structural conflict in its part $ISj/Ci$, and it has necessarily to hit also $Ci \cap ISj$. In fact, by including a component $ci \in Ci \cap ISj$, $d$ assumes that such a faulty component is the cause for the incorrect output $oi$, and, by including a component $cj \in ISj/Ci$, $d$ assumes that such a faulty component masks the fault in $ci$ as far as output $oj$ is concerned. All the incorrect outputs $oh$ other than $oi$ are caused by a faulty component $ch \in Ch/ \cup IS$, that is, by a component that influences output $ch$ only. By hitting the $ISj/Ci$ part of a non-structural conflict, $d$ hits such a non-structural conflict; by hitting $Ci \cap ISj$, $d$ hits structural conflict $Ci$; by hitting each difference set $Ch/ \cup IS$, $d$ hits both each structural conflict $Ch$ and all the non-structural conflicts that include a $Ch/ISr$ part (where $Ch \cap ISr \neq \emptyset$), that is, $d$ hits all the remaining structural and non-structural conflicts. Therefore $d$ is a minimal diagnosis.

**Example 5** Given $OBS(T) = \{KO\ o1, OK\ o2\}$, the only non-structural conflict of system $T$ is $C1/IS2 \cup IS2/C1 = \{M1, A1, A2, M3\}$, where $C1 = \{M1, M2, A1\}$ is the structural conflict inherent to $o1$, and $IS2 = \{M2, M3, A2\}$ is the internal subset inherent to $o2$. There are two diagnoses with masking phenomena, $\{M3, M2\}$ and $\{A2, M2\}$, each being a hitting set of $IS2/C1$ and $C1 \cap IS2$, and having an intersection with $IS2$ whose cardinality is 2.

**Proposition.** A minimal diagnosis without masking phenomena is the minimal hitting set of the difference sets between each structural conflict and all the internal subsets inherent to correct outputs.

**Proof (sketch).** A (minimal) diagnosis $d$ without masking phenomena, being a diagnosis, is a (minimal) hitting set between all (structural and non-structural) conflicts. However, since $d$ is without masking phenomena, it does not include any component belonging to an internal subset inherent to a correct output: thus diagnosis $d$ necessarily hits each non-structural conflict in its part $Ci/ISj$. Therefore, diagnosis $d$ hits all $Ci/ \cup_j ISj$, where $\cup_j ISj$ is the union of all the internal subsets relevant to correct outputs. As such, $d$ is a hitting set also of the sets $Ci/ \cup_j ISj$.

**Corollary.** The diagnosis method adopted in Example 4 and in the remaining of this paper computes minimal diagnoses without masking phenomena.

## 5 DISTRIBUTED DIAGNOSIS

The distributed approach assumes that, at the beginning of each diagnosis session, each diagnoser is supplied with the compiled knowledge (the unmarked matrices) relevant (only) to the process clusters to be considered, which is important if the process contains several mutually exclusive clusters. First a setup is carried out, wherein each local diagnoser annotates its matrices based on the given observation. Afterwards, information about the annotation of matrices is possibly propagated between diagnosers. A local diagnosis phase ensues. Once local candidate diagnoses have been computed, information about them is possibly exchanged between diagnosers. Finally, global candidate diagnoses are drawn. The method is quite flexible, the only standpoints being the setup, the local diagnosis computation, and the global diagnosis computation. Information exchange between diagnosers is not mandatory: either or neither or both exchange phases may be performed.

### 5.1 Setup

Each diagnoser takes into account the projection of the system observation on the relevant process clusters, which, however, does not include any information about the correctness of the process inputs that are the outputs of other processes. Thus, the matrix relevant to a cluster has to include also a column for each input which is not a system input, since the possible incorrectness of such an input, due to a fault in another process, may be the cause of some incorrect outputs of the current cluster.

**Example 6** In Fig. 2, system $T$ is split into three processes, $P1$, $P2$, and $P3$, respectively, where both $P1$ and $P2$ include just one cluster, while $P3$ includes two clusters, say $P32$ and $P33$, the former encompassing component $M2$ only and the latter component $M3$.

Let us cope with the diagnosis problem where the observation (already considered in Example 4) is $OBS(T) = \{KO\ o1,\ OK\ o2\}$, whose projections on the three processes are $OBS(P1) = \{KO\ o1\}$, $OBS(P2) = \{OK\ o2\}$, and $OBS(P3) = \emptyset$. The annotations of the matrices of all clusters, after setup has been completed, are shown in Tables 4, 5 and 6. Matrices $P32$ and $P33$ are completely unmarked since diagnoser $P3$ has no knowledge about the correctness of the relevant outputs ($m2$ and $m3$). Diagnoser $P2$, instead, has a complete knowledge about the relevant output, and, consequently, also of its inputs. Matrices $P1$ and $P2$ are oriented also to the diagnosis of inputs that are not system inputs, that is, $m2$ for $P1$ and both $m2$ and $m3$ for $P2$.

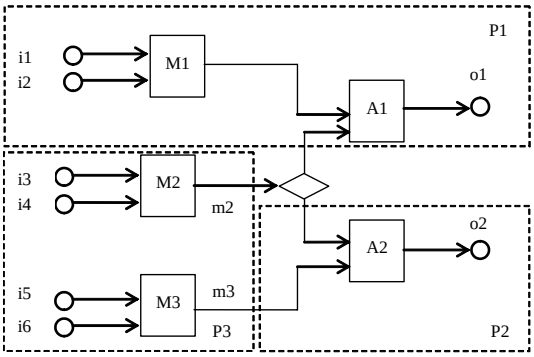


Figure 2: System $T$ split into three processes

### 5.2 Propagation

In this phase, pieces of information inherent to shared variables marked as $OK$ are exchanged between diagnosers. A *shared variable* is an input of a process,

Table 4: Annotated matrix for process $P1$

| | | $A1$ | $M1$ | $m2$ |
|---|---|---|---|---|
| $KO$ | $o1$ | $\surd$ | $\surd$ | $\surd$ |

Table 5: Annotated matrix for process $P2$

| | | $OK$ $A2$ | $OK$ $m2$ | $OK$ $m3$ |
|---|---|---|---|---|
| $OK$ | $o2$ | $\surd$ | $\surd$ | $\surd$ |

called the *predecessor* process, that is the output of another, called the *successor* process. A process may be both a predecessor and a successor of another. Conversely, a process may be neither a predecessor nor a successor of another, in which case the two processes are not *neighbor*. There may be no propagation steps or several ones, since each diagnoser progressively updates its session knowledge based on the pieces of information it receives, and any update may cause in turn another propagation step. The number of steps is finite owing to the finite number of shared variables of the whole system. Three propagation versions are supported: backward, forward and back-and-forth.

**Backward propagation**
According to backward propagation, if, in the setup or in any propagation step, a diagnoser has marked an input of its (which is a shared variable) as $OK$, then it has to notify it to the diagnoser of the predecessor process that generated such an input.

**Example 7** In system $T$, variable $m2$ is shared among all processes, being an output of $P3$ and an input of both $P1$ and $P2$. Variable $m3$ is instead shared between $P2$ and $P3$ only. Let us resume the distributed method to solve the diagnosis problem of Example 6 by performing a backward propagation. Since, in the setup phase, diagnoser $P2$ has marked both $m2$ and $m3$ as $OK$, it notifies it to diagnoser $P3$ (that is, the diagnoser of the predecessor process that generated such shared variables). Hence, diagnoser $P3$ annotates its matrices as shown in Table 7. This finalizes the propagation phase since there are no more pieces of information to be backward propagated.

**Forward propagation**
Dually, according to forward propagation, if, in the setup or in any propagation step, a diagnoser has marked as $OK$ an output of its that is a shared variable, then it has to notify it to the diagnoser of the successor process which is the target of such an output.

**Example 8** Let us consider again the diagnosis problem of Example 6 by performing a forward propagation, as an alternative to the backward propagation described in Example 7. The only process that has any successors is $P3$, therefore only $P3$ can perform a forward propagation. However, given the situation at the end of the setup described in Table 6, no information can be forward propagated since diagnoser $P3$ has no knowledge about the correctness of shared variables ($m2$ and $m3$). Thus, forward propagation ends without adding any further annotation to the matrices.

Table 6: Matrices $P32$ and $P33$

| | $M2$ |
|---|---|
| $m2$ | $\surd$ |

| | $M3$ |
|---|---|
| $m3$ | $\surd$ |

Table 7: Annotated matrices for process $P3$

| | | $OK$ $M2$ |
|---|---|---|
| $OK$ | $m2$ | $\surd$ |

| | | $OK$ $M3$ |
|---|---|---|
| $OK$ | $m3$ | $\surd$ |

**Back-and-forth propagation**
Back-and-forth propagation allows information about shared variables that have been marked as $OK$ to be exchanged between neighbor diagnosers in both directions, that is, a diagnoser can send/receive information to/from the diagnosers of both successor and predecessor processes.

**Example 9** Let us continue the distributed solution of the diagnosis problem of Example 6 by performing a back-and-forth propagation, as an alternative to both the backward propagation in Example 7 and the forward propagation in Example 8. Within back-and-forth propagation, first the backward propagation step described in Example 7 is performed (since it is the only feasible step), matrices $P32$ and $P33$ thus reaching the situation depicted in Table 7. Since, in this step, diagnoser $P3$ has marked row $m2$ of matrix $P32$ as $OK$, it has to notify diagnoser $P1$ (the target process of $m2$) that $m2$ is $OK$, which brings diagnoser $P1$ to update its matrix (that is, Table 4) as shown in Table 8.

Table 8: Matrix $P1$ after back-and-forth propagation

| | | $A1$ | $M1$ | $OK$ $m2$ |
|---|---|---|---|---|
| $KO$ | $o1$ | $\surd$ | $\surd$ | $\surd$ |

### 5.3 Local diagnosis

The phase of local diagnosis applies to each cluster and it is basically carried out as explained in Section 4 for a single cluster that equals the whole system. However, when a cluster does not equal the whole system, also the diagnosis of input shared variables has to be encompassed. Every cluster candidate diagnosis has to be consistent with the given cluster observation as well as with the marked cluster inputs and outputs. Since a diagnoser does not know whether an unmarked process input/output is either correct or incorrect, either

assumptions that it is $OK$ and that it is $KO$ have to be made. Different assumptions lead to different sets of (minimal) cluster candidate diagnoses. A set of cluster candidates of a successor process is *unbound* if it depends on a set of unknown candidates of a predecessor owing to shared variables.

**Example 10** Let us consider the local diagnosis phase following the forward propagation phase in Example 8. Local candidate diagnoses are computed based on Tables 4, 5, and 6, whose situation is the same as the one at the end of the setup. Diagnoser $P1$ draws (from Table 4) two sets of candidate diagnoses (consistent with the given observation $OBS(P1) = \{KO\ o1\}$):
$candidates(P1; KO\ o1, OK\ m2) = \{\{A1\}, \{M1\}\}$,

$$candidates(P1; KO\ \ o1, KO\ \ m2) = \quad (1)$$
$$candidates(P3; KO\ m2).$$

The set of candidates diagnoses in Eq. (1), which is relevant to observation $OBS(P1) = \{KO\ o1, KO\ m2\}$, is unbound: it equals the unknown set(s) of candidate diagnoses inherent to observation(s) of $P3$ (that is, of the predecessor process that generates $m2$) consistent with $OBS(P3) = \{KO\ m2\}$.

Diagnoser $P2$, since all its relevant outputs are $OK$, draws a set of candidate diagnoses that includes the empty diagnosis only, that is,
$candidates(P2; OK\ o2, OK\ m2, OK\ m3) = \{\emptyset\}$.

Diagnoser $P3$ generates the following sets of diagnoses:
$candidates(P32; OK\ m2) = \{\emptyset\}$,
$candidates(P32; KO\ m2) = \{\{M2\}\}$,
$candidates(P33; OK\ m3) = \{\emptyset\}$,
$candidates(P33; KO\ m3) = \{\{M3\}\}$.

**Example 11** Let us consider the local diagnosis phase following the backward propagation phase in Example 7. Local candidate diagnoses are computed based on Tables 4, 5, and 7. Only the situation of diagnoser $P3$ is different with respect to Example 10, therefore $P1$ and $P2$ draw the same local candidate diagnoses as in Example 10, while $P3$ draws
$candidates(P32; OK\ m2) = \{\emptyset\}$,
$candidates(P33; OK\ m3) = \{\emptyset\}$.

**Proposition.** A local diagnosis phase carried out after a back-and-forth propagation leads to globally consistent minimal local diagnoses.

A global diagnosis is monolithically computed based on the system matrix, after all the activities that belong to internal subsets relevant to $OK$ outputs have been marked as $OK$. In a distributed setting, local diagnoses are globally consistent if the same activities as above along with the shared variables that are their input/output flows have been annotated as $OK$ within the cluster matrices. This is achieved by a back-and-forth propagation. In fact, backward steps reach all the 'ancestors' of the processes containing $OK$ system outputs, while forward steps reach all the remaining processes.

### 5.4 Candidate exchange

Information about local candidate diagnoses may now be exchanged between neighbor diagnosers according to either a push or a pull mode. The aim of the current phase is binding the unbound sets of candidates. Whichever the version adopted for the propagation phase, and even if such a phase has been skipped, there may be unbound sets of local candidate diagnoses. Binding them means replacing each of the RHS of equations such as Eq. (1) with a known set of candidate diagnoses (that is, a set of sets of faulty components), and properly updating the LHS so as it reflects both the observation of the current cluster and that of the cluster that actually generated the non-local RHS candidates. This, however, means recording within the session knowledge of a diagnoser some candidate diagnoses produced by another. There may be several sets of candidate diagnoses that can bind an unbound set: all the distinct bindings have to be accomplished.

**Push mode**
The push mode comes in two forms: backward and forward. In the *push backward* mode, the unbound candidates of a successor process are pushed to a predecessor process, so as the predecessor can bind them. A set of cluster candidate diagnoses, whose assumption about a cluster output which is a shared variable is $KO$, is instead pushed from a predecessor to a successor to which such an output is directed if the *push forward* mode is adopted. Such transmitted candidates are those that can be used by the successor diagnoser to bind possibly unbound diagnoses of its.

**Example 12** Let us consider the local diagnoses as computed in Example 10, and let us suppose that a push backward is now performed. The only diagnoser having a set of unbound diagnoses is $P1$, which removes the following set of candidates from its own knowledge and pushes it to diagnoser $P3$:
$candidates(P1; KO\ o1, KO\ m2) = candidates(P3; KO\ m2)$.

Diagnoser $P3$, based on its local knowledge that $candidates(P32; KO\ m2) = \{\{M2\}\}$, draws that
$candidates(P1, P32; KO\ o1, KO\ m2) = \{\{M2\}\}$,

which is added to its previously computed local diagnoses. No further push backward step is feasible.

**Pull mode**
The pull mode supports candidate exchange between neighbor diagnosers only on demand. In other words, each successor diagnoser pulls the candidates it needs from its predecessor diagnosers. The pull mode is based on a bidirectional communication between pairs of neighbor diagnosers since each diagnoser can both receive a request and reply to it. A pull mode session can either be triggered by all the diagnosers that have some unbound sets of local candidates or only by the diagnosers among them that are inherent to processes having incorrect system outputs.

**Example 13** Let us now apply the pull mode after local candidate diagnoses have been computed as in Example 11. The only diagnoser having an unbound set of diagnoses, this being that in Eq. (1), is $P1$ (which is also the only process having incorrect system outputs). Therefore diagnoser $P1$ asks diagnoser $P3$ for $candidates(P3; KO\ m2)$. Diagnoser $P3$, since it has no set of candidate diagnoses for observation $KO\ m2$, replies that $candidates(P3; KO\ m2) = \emptyset$. Hence, diagnoser $P1$ should update its own unbound set of local diagnoses as follows:

$candidates(P1; KO\ o1, KO\ m2) = \emptyset$.

This, however, means removing equality (1) from the sets of candidate diagnoses of $P1$, since an empty set of candidate diagnoses denotes the inconsistency of the assumptions (in this case, the inconsistency of $OBS(P1) = \{KO\ o1, KO\ m2\}$).

### 5.5 Global diagnosis

Whichever the method followed to arrive at this phase, each set of global diagnoses can be computed as the hitting set of consistent sets of candidate diagnoses of all the clusters. A pair of sets of candidate diagnoses is consistent if there is no logical conflict in their assumptions (the bracketed observation). If the pull mode has been applied, global diagnoses can simply be computed as the hitting sets of consistent sets of local diagnoses inherent only to clusters having incorrect system outputs.

**Example 14** Let us now compute global candidate diagnoses at the end of the push backward steps of Example 12. The only consistent sets of local candidate diagnoses of all clusters are
$candidates(P1; KO\ o1, OK\ m2) = \{\{A1\}, \{M1\}\}$,
$candidates(P2; OK\ o2, OK\ m2, OK\ m3) = \{\emptyset\}$,
$candidates(P32; OK\ m2) = \{\emptyset\}$,
$candidates(P33; OK\ m3) = \{\emptyset\}$,

whose hitting set is the same set of global candidate diagnoses obtained in Example 4, that is
$candidates(P1, P2, P32, P33; KO\ o1, OK\ o2, OK\ m2, OKm3)$ = $\{\{A1\}, \{M1\}\}$.

**Example 15** The sets of global diagnoses, after the pull mode in Example 13 has been applied, can be computed as the hitting sets of the consistent sets of local diagnoses of clusters belonging to processes having incorrect system outputs. There is just one such cluster, $P1$, whose only left set of local diagnoses is $candidates(P1; KO\ o1, OK\ m2) = \{\{A1\}, \{M1\}\}$, which therefore are also global diagnoses, where the observation is implicitly $OK$ for all the system outputs of the processes that are not mentioned in the candidate assumptions.

## 6 CONCLUSION

This paper is an attempt to apply consistency-based diagnosis both in an abstract and in a distributed way, the former meaning that no component behavioral model is adopted, the latter that distinct reasoning agents, the diagnosers, are in charge of computing the local diagnoses of arbitrary parts, called processes, of the considered system. Each diagnoser is provided with compiled knowledge inherent to the execution paths of the relevant process. There is not just one method for achieving the diagnostic results but several methods since three mandatory phases are interleaved with two optional ones, and each optional phase can be performed in several different ways. The specific versions (if any) adopted for the optional phases affect the accomplishment of the last phase, that is, the computation of global diagnoses. This consists in the computation, possibly performed by an additional reasoning agent, of the hitting sets of mutually consistent sets of local candidate diagnoses. Some interesting outcomes are listed here below.

- Global diagnoses can be obtained even if no information exchange has taken place between diagnosers.
- In case there is just one process having incorrect system outputs (as, for instance, all the system outputs have purposely been assigned to just one process), at the end of some specific methods (that necessarily encompass information exchange between neighbor diagnosers) the diagnoser of such a process knows all the global diagnoses, without any need for an additional agent to compute the hitting sets of local diagnoses.
- The nature of the information exchanged between (neighbor) diagnosers in the two (optional) phases of the proposed approach is twofold: information about the correctness of shared variables in the former phase, and information about local candidate diagnoses in the latter. If the former phase is carried out, the amount of information exchanged in the latter phase is smaller than that needed if the former phase is skipped.
- The effort for the final phase of global diagnosis computation depends on the amount of exchanged information: if no information has been exchanged, the larger the number of local candidate diagnoses and the heavier the work to sort out consistent sets of local candidate diagnoses.

The proposed approach has been described along with examples embodying it. A more formal presentation, including the proof of the correctness of the method in computing a sound and complete set of local and global minimal diagnoses without masking phenomena is needed. Future work will extend the distributed approach to the computation of diagnoses with masking phenomena.

## REFERENCES

(Armant *et al.*, 2008) V. Armant, P. Dague, and L. Simon. Distributed consistency-based diagnosis without conflicts. In *19th International Workshop on Principles of Diagnosis –DX'08*, Blue Mountains, AU, 2008.

(Borrego *et al.*, 2009) D. Borrego, R.M. Gasca, M.T. Gómez López, and I. Barba. Choreography analysis for diagnosing faulty activities in business-to-business collaboration. In *20th International Workshop on Principles of Diagnosis – DX'09*, pages 171–178, Stockholm, S, 2009.

(Davis, 1984) R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1):347–410, 1984.

(de Kleer and Williams, 1987) J. de Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

(Fowler, 2004) M. Fowler. *UML Distilled: a brief guide to the standard object modeling language*. Addison-Wesley, 2004.

(Reiter, 1987) R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.