Linköping Studies in Science and Technology
Thesis No. 1176

Distributed Diagnosis and Simulation Based Residual Generators

Jonas Biteus

Linköpings universitet
INSTITUTE OF TECHNOLOGY

Vehicular Systems
Department of Electrical Engineering
Linköpings universitet, SE – 581 83 Linköping, Sweden
E-mail: biteus@isy.liu.se

Linköping 2005

DISTRIBUTED DIAGNOSIS
AND SIMULATION BASED RESIDUAL
GENERATORS

Department of Electrical Engineering
Linköpings Universitet
SE – 581 83 Linköping
Sweden

ISBN 91-85299-73-1
ISSN 0280-7971
LIU-TEK-LIC-2005:31

ABSTRACT

Fault diagnosis is becoming increasingly important for many technical systems. This is for example true in automotive vehicles where fault diagnosis is needed due to economic reasons such as efficient repair and fault prevention, and legislations that mainly deal with safety and pollution. The objective for a diagnostic system is to detect and isolate faults in the system. A diagnostic system consists of several specialized parts, for example residual generators, diagnoses calculation, and communication with other systems.

In embedded systems with dozens of electronic control units that individually states local diagnoses, it can be computationally expensive to find which combination of local diagnoses that points at the correct set of faulty components. A distributed method is proposed where local diagnoses are extended using networked information. The extension is done thru the sharing of local conflicts or local diagnoses between the electronic control units. The number of global diagnoses grows with the number of local diagnoses. Therefore, an algorithm is presented that from the local diagnoses calculates the more likely global diagnoses. This restriction to the more likely diagnoses is sometimes appropriate since there are limitations in processing power, memory, and network capacity.

A common approach to design diagnostic systems is to use residual generators, where each residual generator is sensitive to some faults. A method is presented that constructs residual generators from sets of overdetermined model equations, such that simulation can be used to determine if the residual is zero or not. The method thus avoids the need to analytically transform the set of equations into some specific residual generator form. It can also utilize smaller sub sets of equations like minimally overdetermined sets, and it can further take advantage of object-oriented simulation tools.

Keywords: Diagnosis; Distributed diagnosis; Fault isolation; Residual generator; Simulation.

ACKNOWLEDGMENT

This work was performed at the department of Electrical Engineering, division of Vehicular Systems, Linköpings universitet in Sweden. I would like to thank my professor and supervisor, *Lars Nielsen* for letting me perform this work at the division.

Thanks goes to *Mattias Nyberg* who lead me into the research area of diagnosis, and whom I have performed collaborative work with. To *Erik Frisk* for many fruitful discussions about the topics in the thesis, and for spending many hours proofreading the thesis. To *Jan Åslund* for proofreading the thesis. To *Mattias Krysander* for discussions about diagnosis.

I would also like to thank the staff at Vehicular Systems for creating a nice working atmosphere.

To Scania AB for collaborative projects. To *Mathias Jensen* for discussions about distributed diagnosis. To *Magnus Adolfson* and *David Elfvik* for general discussions about diagnosis in automotive engines.

This work has been supported by The Swedish Foundation for Strategic Research thru the graduate school ECSEL (The Excellence Center in Computer Science and Systems Engineering in Linköping) and the project VISIMOD (visualization, modeling, simulation and system identification).

Jonas Biteus
Linköping, June 2005

CONTENTS

THESIS INTRODUCTION

There are an increasing number of technical systems that use multiple agents to achieve control objectives. One such example is found in the vehicle industry, where new vehicles might include several dozens of *electronic control units* (ECUs), which are used to control different parts of the vehicle. With an increasing complexity of the systems comes a higher demand for fault diagnosis, i.e. to detect and localize faults in the system. The systems therefore include diagnostic systems which are responsible for the detection and isolation of faults. One of the objectives for the diagnostic system is to calculate the diagnoses, where each diagnosis points at some set of possibly faulty components. The diagnoses can be used in for example fault tolerant control, for repair, or to notify the driver as an alarm.

Due to the increasing number of connections between the agents, new demands are put on the diagnostic systems. One such demand is that the agents should be able to communicate with each other to compute diagnoses that are consistent with the knowledge stored in all agents. Part I of this thesis deals with fault isolation in systems that are distributed.

To detect faults, diagnostic systems often include diagnostic tests, where each test is sensitive to some faults. These diagnostic tests can for example be constructed from residual generators, which are based on a model of the system. Part II of this thesis deals with the construction of residual generators.

OUTLINE OF THE THESIS

Outline of Part I.

Chapter 1 Introduction Part I.

Chapter 2 A background to consistency based diagnosis and to distributed diagnosis is presented.

Chapter 3 A framework for distributed diagnosis is presented. The framework starts with a definition of diagnoses suitable to use when considering distributed systems. The diagnoses can include both components and inputs, which are input signals from other agents. A system description for distributed systems is then defined. Since the diagnoses can include inputs that depends on other components and inputs, there is a possibility that a fault in one component propagates thru inputs to many other diagnoses. The effects of these propagations are studied.

In a distributed system, there are local conflicts and local diagnoses, i.e. conflicts and diagnoses in one agent, and global conflicts and global diagnoses, i.e. conflicts and diagnoses for the complete system. The relations between local conflicts, local diagnoses, global conflicts, and global diagnoses are described.

The relation between more likely diagnoses and the number of components included in the diagnoses is presented for distributed diagnoses. It is described how the global diagnoses can be divided into disjoint parts of diagnoses, denoted module diagnoses. Since these parts are smaller than the global diagnoses, they are more easily understood for a technician.

Chapter 4 It is shown how local conflicts and local diagnoses can be shared between the agents. The primary gain from this sharing of information is that an agent can state more complete diagnoses about the components used in the agent, than it could do if it only used its own local conflicts. Thus, the local diagnoses can be extended as a result of the sharing of information.

The decision of which conflicts to send is based either on the local conflicts or the local diagnoses. To reduce the complexity of the extended diagnoses, it is shown how the size of the conflicts and the diagnoses can be reduced while preserving the consistency of the local conflicts and local diagnoses.

Chapter 5 The chapter presents algorithms that use the methods described in Chapter 4. Algorithms for extending local diagnoses by sharing conflicts and diagnoses are presented. The algorithms

can base their choice of which conflicts to transmit on all local diagnoses, or only on the more likely local diagnoses.

The feasibility for the different algorithms are shown for some simulations. The simulations are based on a hypothetical model of an embedded system that is inspired by an existing system. In the model, components, inputs, outputs, diagnostic tests, and faults are picked by random.

Chapter 6 An algorithm is presented that from local diagnoses calculates all global diagnoses with minimal cardinality, i.e. the global diagnoses with the least number of components. The computation can be performed centralized or it can be distributed over the agents. The algorithm partitions the diagnoses into sets of disjoint diagnoses, thereby further limiting the complexity. The algorithm is efficient and it only needs a small number of transmissions over the network to calculate the diagnoses.

A similar model to the one used in the preceding chapter is used for simulations. The feasibility and calculation times for the algorithm are shown for some simulations.

Chapter 7 Conclusions Part I.

Outline of Part II.

Chapter 8 Introduction Part II.

Chapter 9 In this chapter, a method for the construction of residual generators is described. The residual generators are constructed from smaller sub sets of model equations. These sets are overdetermined such that there exist exactly one more equation than unknown variables in the set of equations.

By adding a residual variable to the set of equations, an exactly determined set of equations is constructed. It is shown how the residual variable can be added such that the set of equations can be simulated. The method avoids the need to analytically transform the sets of equations into some specific residual generator form.

It is shown how it is possible to choose between either a more complex residual generator or the need to perform derivative approximations of sensor values.

Chapter 10 The method presented in the preceding chapter is exemplified on a model for a non-linear point-mass satellite. It is shown how the addition of approximations of derivatives of sensor values results in different residual generators.

Chapter 11 It is shown how the method can take advantage of the simulation tool Dymola which use the simulation language Modelica to extract and simulate the set of equations.

Chapter 12 Conclusions Part II.

CONTRIBUTIONS OF THE THESIS

The main contributions of the thesis are:

- The framework useful for consistency based distributed diagnosis, including for example module diagnoses. Introduced in Chapter 3.
- The description of how local diagnoses can be extended thru the sharing of conflicts or local diagnoses, where the choice of which conflicts to share could be based either on the conflicts or the local diagnoses. Introduced in Chapter 4.
- The algorithms for the sharing of conflicts and diagnoses. Introduced in Chapter 5.
- The algorithm that calculates all minimal cardinality module diagnoses. Introduced in Chapter 6.
- The description of how residual generators can be designed such that simulation is used to calculate the value of the residual, and thereby avoiding the need to analytically transform the residual generators into some specific residual generator form. Introduced in Chapter 9.
- The description of how DAE simulators like Dymola, which use the modeling language Modelica, could be utilized to simulate residual generators. Introduced in Chapter 11.

PUBLICATIONS

Some of the material presented in this thesis has been published in the following papers.

- [BJN04] – In this paper, a first approach that could be used for finding minimal cardinality diagnoses in a distributed environment was presented. The material is here presented in Chapter 6.
- [BJN05] – In this paper, the result in [BJN04] was improved.

- [BN02] – This paper presented ideas about how residual generators could be constructed and evaluated with simulation tools.
- [BN03] – A continuation and improvement of the methods presented in [BN02].

Material by the same author that relates to diagnosis but has not been included in this thesis.

- [Bit02] – A mean value model of a heavy-duty engine was presented. The model could for example be used for diagnosis or control.
- [BCF+04] – How aircraft safety could be increased by the introduction of FDI methods was discussed in this report.
- [ÅBF+05] – This paper described a systematic inclusion of diagnosis performance in fault tree analysis.

INTRODUCTION TO FAULT DIAGNOSIS

This introduction is an extraction from [NF05]. It has been used at the division of Vehicular System to introduce fault diagnosis.

What is Diagnosis?

From a general perspective, including both the medical and technical case, diagnosis can be explained as follows. For a process, there are observed variables or behaviors for which there is knowledge of what is expected or normal. The task of diagnosis is to, from the observations and the knowledge, generate a diagnosis, i.e. to decide whether there is a fault or not and also to identify the fault. Thus, the basic problems in the area of diagnosis are how the procedure for generating diagnoses should look like, what variables or behaviors that are relevant to study, and how to derive the knowledge of what is expected or normal.

This introduction focuses on diagnosis of technical systems and the goal is to find malfunctions in for example sensors and actuators. The observations are mainly signals obtained from the sensors, but can also be observations made by a human. Examples of such human observations are for example level of noise or vibrations. The diagnosis is computed by observing inconsistencies between observed variables and what is considered to be normal behavior. If the diagnosis is based on an explicit model of the system, then the term model-based diagnosis is used. Diagnosis of technical systems can be performed off-line or

on-line. If on-line diagnosis is considered, then the diagnosis is usually automated so it is performed without involvement of humans. Most concepts described in here are equally applicable to off-line and on-line diagnosis.

The Use of Diagnosis

Diagnosis systems have found their way into many applications. In the context of model-based diagnosis, some important areas that have been discussed in the literature are:

- Nearly all subsystems of aircrafts, e.g. aircraft control systems, navigation systems, and engines.
- Emission control systems in automotive vehicles.
- Nuclear plants.
- Chemical plants.
- Gas turbines.
- Industrial robots.
- Electrical motors.

For these systems and also for technical processes in general, the main reasons to incorporate diagnostic systems are:

Safety In many technical systems a fault may cause serious personal damage. This is especially obvious in safety critical processes such as aircrafts and nuclear plants. For these systems, high reliability and security of the system is fundamental.

Environment Protection In for example emission control systems in automotive vehicles, a fault may cause increased emissions. It has been concluded that a major part of the total emissions from cars originates from vehicles with malfunctioning emission control systems. Other important examples are nuclear plants and chemical plants in which a fault may cause serious damage to the environment.

Machine Protection A fault can often cause damage to the machine. Therefore, it is important that faults are detected as quickly as possible after they have occurred.

Availability For many technical systems it is critical that the systems are running continuously. This is for example the case for gas turbines in power plants and industrial robots. The reasons may be economical as well as safety. With the help of a diagnostic system, early warnings can be obtained before serious breakdown. When the fault has been detected, the system can be stopped until repair or rather be switched into a new mode. In the new mode, the performance of the system may be degraded but at least more serious breakdowns can be avoided.

Reparability Closely connected to availability is reparability. A good diagnostic system will quickly identify the faulty component that should be replaced. In this way, time-consuming fault localization is reduced, which will decrease total repair time.

Flexible Maintenance Maintenance can be expensive since the machine or process often needs to be taken out of operation. Therefore, it is desirable to make sure that the machine is not taken out of operation for maintenance when there is no need for maintenance. It is also desirable to be able to plan maintenance stops in advance to be able to disturb the production as little as possible. A diagnostic system that detects faults early, desirably before more serious faults occur, can hopefully help both to avoid unnecessary maintenance and to indicate far in advance when maintenance is needed.

A Short History

Manual diagnosis has been performed as long as there have existed technical systems, but automatic diagnosis started to appear first when computers became available. In the beginning of the 70's, the first research reports on model-based diagnosis were published. Some of the earliest areas that were investigated were chemical plants and aerospace applications. The research on model-based diagnosis has since then been intensified during both the 80's and the 90's. Today, this is still an expansive research area. Up to now, numerous methods for doing diagnosis have been published. Unfortunately, many approaches are more ad hoc than systematic and it is fair to say that few general theories exist and there is not yet a complete understanding of the relations between different methods. This is reflected in the shortage of books in the area and the fact that no general terminology has yet been agreed upon. However, the importance of diagnosis is unquestioned. This can be exemplified by the computerized management systems for automotive engines, used to control the engine. For these systems, more than 50 % of the software can nowadays be dedicated to diagnosis. The rest is for example for control.

I

Distributed Diagnosis

1

Introduction Part I

Most modern automotive vehicles include several ECUs which communicate over an electronic network. Each ECU is usually connected to one or several *components*, e.g. sensors and actuators, and to make sure that the components are operating correctly, they are monitored by the ECUs. For a deeper discussion about distributed systems in vehicle applications, see for example [HVL05, Chapter "Vehicle Application of Controller Area Network"].

The ECUs connected to a network is one example of a distributed system. In more general terms, each ECU can be denoted an *agent* which is a more or less independent software entity. Distributed systems are characterized by being partitioned into several agents, where each agent only have a partial knowledge of the system. For an introduction to agents, see for example [Hay99].

Often *diagnostic tests*, which can be simple or complex, are used to perform the monitoring. The use of the diagnostic tests results in *conflicts*, i.e. a set of components that can not all be non-faulty. The conflicts can be used in different ways. The most direct approach is to collect all conflicts in a central unit and then use the results to calculate the *global diagnoses*, where each global diagnosis states a diagnosis for the complete system. Since each global diagnosis is a set of possibly faulty components, they are easy to understand. Unfortunately, the number of global diagnoses grows exponentially both with the number and the size of the conflicts.

An alternative is to directly calculate *local diagnoses* from the conflicts in each agent. Since the local diagnoses from different agents might include the same components, a combination of local diagnoses must be used to correctly calculate each global diagnosis.

How to use these local diagnoses to calculate more complete local diagnoses and how to calculate the more likely global diagnoses are the main topics of this part of the thesis.

Extending Local Diagnoses

By transmitting conflicts or diagnoses between agents, it is possible to extend the local diagnoses. The agent receiving the transmitted information might therefore gain more complete local diagnoses.

The primary gain from this sharing of diagnostic information is that an agent can state more complete diagnoses about its own diagnosable objects than it could do if it only used its own local conflicts. Thus, the local diagnoses can be *extended* as a result of the sharing of diagnostic information.

Local Diagnoses to More Likely Global Diagnoses

The global diagnoses can be found by merging all local diagnoses with each other. This might however lead to an exponential growth of the number of global diagnoses. To reduce the growth of this combinatorial explosion, it is sometimes useful to only consider the diagnoses that are more likely to be correct. Which diagnoses that are more likely could be decided in several different ways. In this thesis, the diagnoses with the lowest *cardinality*, i.e. the lowest number of faulty components, will be considered to be the most likely diagnosis.

An algorithm is presented that calculates the global diagnoses with *minimal cardinality* from the local diagnoses. To reduce the complexity, the global diagnoses are partitioned into smaller sets of diagnoses, where each of these sets is guaranteed to be free of complex relations to the others.

The algorithm can be run in a central diagnostic computer, or it can distribute the computation intense tasks to the local agents. Often, there are limitations in both processing power, memory, and network capacity, therefore, these possibilities makes the algorithm more versatile.

1.1 A TYPICAL DISTRIBUTED SYSTEM

Many vehicles have a *controller area network* (CAN) which connects several ECUs to each other. Figure 1.1 shows a configuration of the embed-

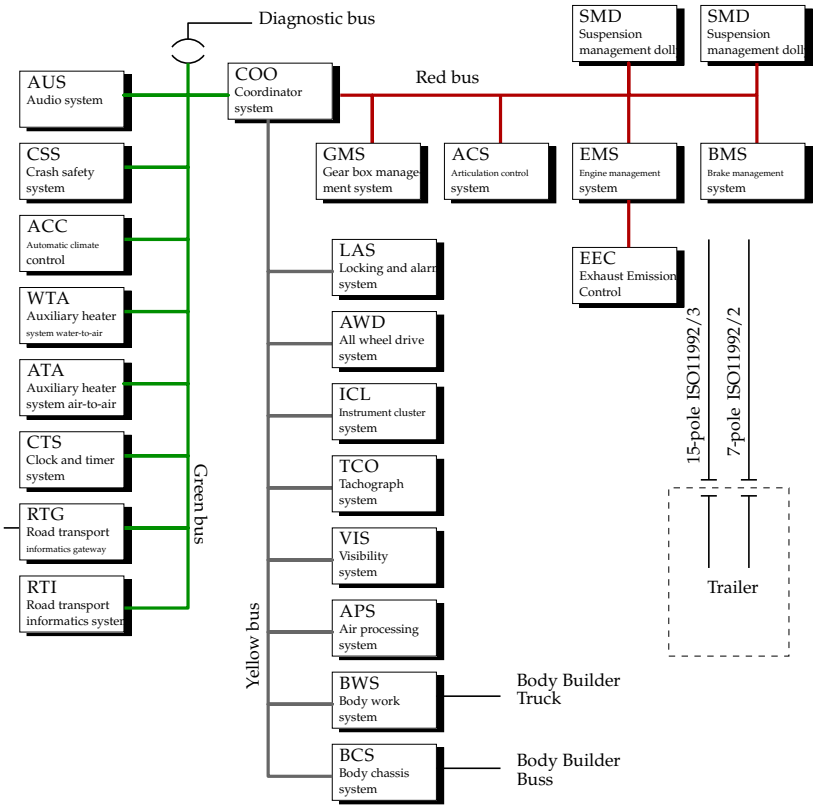


FIGURE 1.1: Embedded system in current Scania heavy-duty vehicles.

ded system used in the current Scania heavy-duty vehicles. It includes three separate CAN buses, red, yellow, and green, which are connected to the coordinator ECU. The coordinator ECU acts as a router, making sure that no message is forwarded to any other bus unless it is necessary. Each of the ECUs is further connected to sensors and actuators, and both sensor values and control signals can be shared with the other ECUs over the network. An example of an ECU is the engine management system, which is connected to sensors and actuators related to the engine.

There are between 20 and 30 ECUs in the system, depending on the type and outfit of the truck, and between 4 and 110 components are connected to each ECU. The ECUs' CPUs have a clocking speed of 8 to 64 MHz, and a memory capacity of 4 to 150 kB. The current Scania diagnostic system consists of precompiled diagnostic tests and there

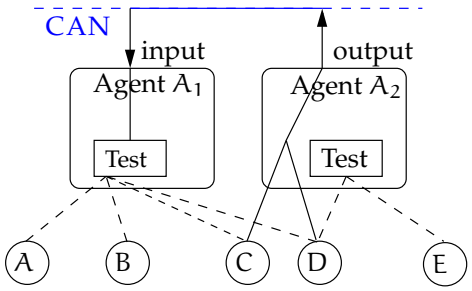


FIGURE 1.2: A typical ECU, component and test layout.

are between 10 and 1000 tests in each ECU.

EXAMPLE 1.1: Figure 1.2 shows a typical layout of ECUs and components. The system consists of two agents, five components, and a CAN network. The test in agent A_2 involve the components D and E, which are connected with dashed lines. The test in A_2 involve components A to D. Some calculated value used in a test in agent A_1 that involves component C and D is transmitted over the network from agent A_1 to A_2. Component C and D might for example be sensors that are physically connected to agent A_2. ◇

1.2 OUTLINE OF PART I

The outline was given in the thesis introduction and are repeated here for completeness.

Chapter 1 Introduction Part I.

Chapter 2 A background to consistency based diagnosis and to distributed diagnosis is presented.

Chapter 3 A framework for distributed diagnosis is presented. The framework starts with a definition of diagnoses suitable to use when considering distributed systems. The diagnoses can include both components and inputs, which are input signals from other agents. A system description for distributed systems is then defined. Since the diagnoses can include inputs that depends on other components and inputs, there is a possibility that a fault in one component propagates thru inputs to many other diagnoses. The effects of these propagations are studied.

In a distributed system, there are local conflicts and local diagnoses, i.e. conflicts and diagnoses in one agent, and global con-

flicts and global diagnoses, i.e. conflicts and diagnoses for the complete system. The relations between local conflicts, local diagnoses, global conflicts, and global diagnoses are described.

The relation between more likely diagnoses and the number of components included in the diagnoses is presented for distributed diagnoses. It is described how the global diagnoses can be divided into disjoint parts of diagnoses, denoted module diagnoses. Since these parts are smaller than the global diagnoses, they are more easily understood for a technician.

Chapter 4 It is shown how local conflicts and local diagnoses can be shared between the agents. The primary gain from this sharing of information is that an agent can state more complete diagnoses about the components used in the agent, than it could do if it only used its own local conflicts. Thus, the local diagnoses can be extended as a result of the sharing of information.

The decision of which conflicts to send is based either on the local conflicts or the local diagnoses. To reduce the complexity of the extended diagnoses, it is shown how the size of the conflicts and the diagnoses can be reduced while preserving the consistency of the local conflicts and local diagnoses.

Chapter 5 The chapter presents algorithms that use the methods described in Chapter 4. Algorithms for extending local diagnoses by sharing conflicts and diagnoses are presented. The algorithms can base their choice of which conflicts to transmit on all local diagnoses, or only on the more likely local diagnoses.

The feasibility for the different algorithms are shown for some simulations. The simulations are based on a hypothetical model of an embedded system that is inspired by an existing system. In the model, components, inputs, outputs, diagnostic tests, and faults are picked by random.

Chapter 6 An algorithm is presented that from local diagnoses calculates all global diagnoses with minimal cardinality, i.e. the global diagnoses with the least number of components. The computation can be performed centralized or it can be distributed over the agents. The algorithm partitions the diagnoses into sets of disjoint diagnoses, thereby further limiting the complexity. The algorithm is efficient and it only needs a small number of transmissions over the network to calculate the diagnoses.

A similar model to the one used in the preceding chapter is used for simulations. The feasibility and calculation times for the algorithm are shown for some simulations.

Chapter 7 Conclusions Part I.

1.3 RELATED WORK

The concept of Agents have been studied in the academic area for quite some time, and has also started to emerge in the industry. An overview of how software agents can be used in industry is given in [Par98].

Diagnosis for embedded systems can be centralized or distributed. Most research has been aimed at the centralized problem, where a single process collects relevant data from the system and calculates global diagnoses. The fundamental paper [Rei87] studies the problem of generating consistent diagnoses from a set of conflicts. In contrast to the centralized system there is no central process in a fully distributed system. The distributed processes instead communicate with each other to form the global diagnoses.

Distributed diagnosis has mostly been discussed for discrete event dynamic systems, see for example [LS01]. One paper that discusses distributed consistency based diagnosis is [RTW03], where an algorithm is presented which can be used for distributed diagnosis. The algorithm uses both consistency and abduction based diagnosis, which is another type of diagnosis. The algorithm stipulates how conflicts should be exchanged so that each agent can state global diagnoses.

The standard formulation in model-based diagnosis, as used in the *artificial intelligence* (AI) community, is used in for example [RTW03] to describe a distributed diagnostic system. Also in [KKZ02] a distributed diagnosis framework is presented. The paper discusses both global and local diagnoses.

The paper [Pro02] presents a distributed model-based diagnostics architecture for embedded diagnostics. The traditional model-based definitions of diagnosis, as used in the AI community, is extended to a distributed environment. Each component, in a set of components, can obtain a single diagnosis, the algorithms then combines these diagnosis to obtain the global diagnoses.

A diagnostic system does not work independently from the rest of the system. The complete system includes protocols for network communication, control algorithms, and much more. In this thesis only the algorithms used in the diagnostic systems are primarily considered. Communication protocols and storage handling are for example not discussed in detail. However, the overall network architecture used for communication, storage, etc, in a distributed system is important.

The EU funded project Multi-Agents-based Diagnostic Data Acquisition and Management in Complex Systems (MAGIC) develops such an architecture useful for distributed diagnosis. The project is

presented in for example [KS03]. The aim for their project is to develop a general purpose architecture for diagnosis in complex systems. MAGIC is partially a continuation of the project Distributed Architecture for Monitoring and Diagnosis (DIAMOND). It was introduced in for example [ATL02]. The paper [ALW+03] discusses the concept of an agent as used in the MAGIC project. Such an agent used for diagnosis is presented in [RTF03]. This particular diagnosis agent is based on a hidden Markov model to perform diagnosis.

1.4 PUBLICATIONS

Some of the materials in this part have been published in two conference articles.

- [BJN04] – This paper presented how a merge of local diagnoses could be done so that the result was the minimal cardinality global diagnoses.
- [BJN05] – The above publication was extended in this paper where the partition of the diagnoses was done in a better way. The merge order was also analyzed in more detail.

2

Background to Consistency Based Diagnosis

This chapter will briefly introduce the concept of consistency-based diagnosis. The motivation is not to give a complete introduction to diagnosis, but to introduce the formalism that will be used in the rest of the thesis. A more thorough introduction to consistency based diagnosis can be found in for example the collection [HCK92].

2.1 Consistency Based Diagnosis

A system consists of a set of components, which should be supervised by the diagnostic system. The objective of the diagnostic system is to *detect* and *isolate* the components that are behaving abnormal.

Model based diagnosis compares a *model* of a system with available *observations*. Deviations between the model and the observations can then be used to obtain diagnoses for the system's behavior. Model based diagnosis can be used in conjunction with *consistency based diagnosis*. The objective in consistency based diagnosis is to derive a set of assignments to the components in the model, so that the model, the observations, and the assignments are consistent with each other.

Each component can be in some mode, e.g. the normal, the abnormal, or some specific fault mode. The model itself is said to be in a specific system mode if all components have been assigned some mode. If the model, the observations, and a system mode are inconsistent, it

is concluded that the system mode is in-correct, i.e. the system mode represents a *conflict*. From a set of such conflicts, it is possible to draw the logical implications, which are the *diagnoses*. In other terms, a diagnosis states a possible mode for the system that is consistent with the conflicts.

The model is described by its *system description* SD, which is a set of logical rules describing the behavior of the system. These rules can for example be a set of equations, such as a state-space model. The observations are denoted by OBS, which for example can be a set of sensor and actuator values.

A component is something that can be diagnosed. This not only includes physical things such as pipes, actuators, etc., but it also includes more diffuse things such as the connection between a sensor and a cable.

2.2 BEHAVIORAL MODES

Each component $c \in \mathcal{C}$ has a fault mode AB (abnormal), which does not have a model, a normal mode $\neg AB$, and one or several specific fault modes. If a component $c \in \mathcal{C}$ is in the abnormal mode, then $\mathtt{mode}(AB, c)$ is true. The notation

$$\mathtt{mode}(AB, c) \triangleq AB(c)$$

will be used.

To reduce the complexity of the diagnostic system, it is sometimes preferable to only consider the AB and the $\neg AB$ mode. Further, since it is possible to represent the fault modes with virtual components and preference lattices as shown in [Bre96, Chapter 8], the diagnostic system can be represented in only the fault modes AB and $\neg AB$ with the addition of a "fault-mode lattice". Therefore, from now on the following assumption is made.

ASSUMPTION 2.1 (Minimal diagnosis hypothesis): *A component* $c \in \mathcal{C}$ *can only be in the* AB *and the* $\neg AB$ *mode, where the* AB *mode does not have a model.*

The assumption is, in a different notation, stated in for example the paper [dKMR92]. With this assumption, the notation in for example GDE can be employed [HCK92]. This notation replaces the logical expressions with sets, where the sets are used to represent both conflicts and diagnoses. The following example shows how the notation is used.

EXAMPLE 2.1: If two components A and B are in the faulty mode, this is written in logic form as

$$AB(A) \wedge AB(B)$$

which can be represented by

$$\{A, B\}$$

in the set notation. ◇

2.3 DIAGNOSES

The following definition of diagnosis is often used in consistency based diagnosis

DEFINITION 2.1 (Diagnosis [dKMR92]): *A diagnosis is a set of components* $D \subseteq \mathcal{C}$ *so that*

$$SD \cup OBS \cup \{\bigwedge_{c \in D} AB(c) \wedge \bigwedge_{c \in \mathcal{C} \setminus D} \neg AB(c)\}$$

is consistent.

A diagnosis states a mode assignment to the components included in the system consistent with the system description and the observations. The following corollary shows an important concept often used in diagnosis.

THEOREM 2.1 (Minimal diagnosis hypothesis [dKMR92]): *If Assumption 2.1 is true and* D *is a diagnosis, then all supersets of* D *are also diagnoses.*

Proof. Since AB has no model

$$SD \cup OBS \cup \{\bigwedge_{c \in \bar{D}} AB(c) \wedge \bigwedge_{c \in \mathcal{C} \setminus \bar{D}} \neg AB(c)\}$$

is satisfied for any superset $\bar{D} \supseteq D$. □

Thus, with Assumption 2.1, a superset of a diagnosis is also a diagnosis. This leads to minimal diagnoses.

DEFINITION 2.2 (Minimal diagnosis [dKMR92]): *A diagnosis* D' *is a minimal diagnosis if there is no proper subset*

$$D \subsetneq D'$$

where D *is a diagnosis.*

The set of minimal diagnoses completely characterizes all possible diagnoses, i.e. if the set of minimal diagnoses is known, then the set of all diagnoses is known. As a result of this, the non-minimal diagnoses can safely be removed.

Sometimes it is preferable to only state the exact modes for some parts of the components. For this partial diagnoses can be used.

DEFINITION 2.3 (Partial diagnosis [dKMR92]): *A partial diagnosis is a set of components* $D \subseteq \mathcal{C}$ *so that*

$$SD \cup OBS \cup \{\bigwedge_{c \in D} AB(c) \wedge \bigwedge_{c \in \mathcal{C} \setminus D} A(c)\}$$

is consistent either when $A(c)$ *is* $AB(c)$ *or when it is* $\neg AB(c)$.

The minimal partial diagnoses are denoted kernel diagnoses.

DEFINITION 2.4 (Kernel diagnosis [dKMR92]): *A partial diagnosis* D' *is a kernel diagnosis if there is no proper subset*

$$D \subsetneq D'$$

where D *is a partial diagnosis.*

The minimal diagnoses and the kernel diagnoses have a strong relationship. If Assumption 2.1 is fulfilled, then the minimal diagnoses have a one-to-one relation with the kernel diagnoses [dKMR92]. This is used in the set notation where both type of diagnoses are represented by the same set. Notice however that they are not equal.

EXAMPLE 2.2: Consider the diagnoses

$$(AB(A) \wedge \neg AB(B) \wedge \neg AB(C)) \vee (AB(A) \wedge AB(B) \wedge \neg AB(C))$$

and the partial diagnoses

$$(AB(A) \wedge A(B) \wedge A(C)) \vee (AB(A) \wedge AB(B) \wedge A(C)).$$

Both these sets are represented by the set

$$\{\{A\}, \{A, B\}\}.$$

The minimal diagnosis is

$$AB(A) \wedge \neg AB(B) \wedge \neg AB(C)$$

and the kernel diagnosis is

$$AB(A) \wedge A(B) \wedge A(C)$$

Both these sets are represented by the set

$$\{\{A\}\}.$$

$\diamond$

2.4 CONFLICTS

In most diagnostic systems, the diagnoses are not obtained directly from the model and the observations. The commonly used approach is instead to first find conflicts and then from these conflicts indirectly derive the diagnoses.

DEFINITION 2.5 (Conflict [dKMR92]): *A conflict is a set of components $\pi \subseteq \mathcal{C}$ so that*

$$SD \cup OBS \cup \{\bigwedge_{c \in \pi} \neg AB(c)\}$$

is inconsistent.

A conflict states a possible mode assignment for some set of components, which is inconsistent with the observations and the model. A set of conflicts is denoted Π. From the conflicts follows the minimal conflicts.

DEFINITION 2.6 (Minimal conflicts [dKMR92]): *A conflict π' is a minimal conflict if there is no proper subset*

$$\pi \subsetneq \pi'$$

where π is a conflict.

The set of minimal conflicts completely characterizes all possible conflicts.

2.5 RELATIONS BETWEEN CONFLICTS AND DIAGNOSES

A conflict states that not all components in a set can be in the non-abnormal mode, while a diagnosis states a set of components that are in the abnormal mode. The diagnoses can be seen as the logical implication of the set of conflicts. A useful relation between diagnoses and conflicts is given in the following theorem. It is stated in [dK92, Theorem 1] in a different notation.

THEOREM 2.2 (Conflicts to diagnoses): *Let Π be a set of conflicts. The set $D \subseteq \mathcal{C}$ is a diagnosis if*

$$D \cap \pi \neq \emptyset$$

for all $\pi \in \Pi$.

When using set notation, it is sometimes useful to represent a set of diagnoses with a lattice. Such a lattice representing the diagnoses for five components is shown in Figure 2.1. The bottom node is the

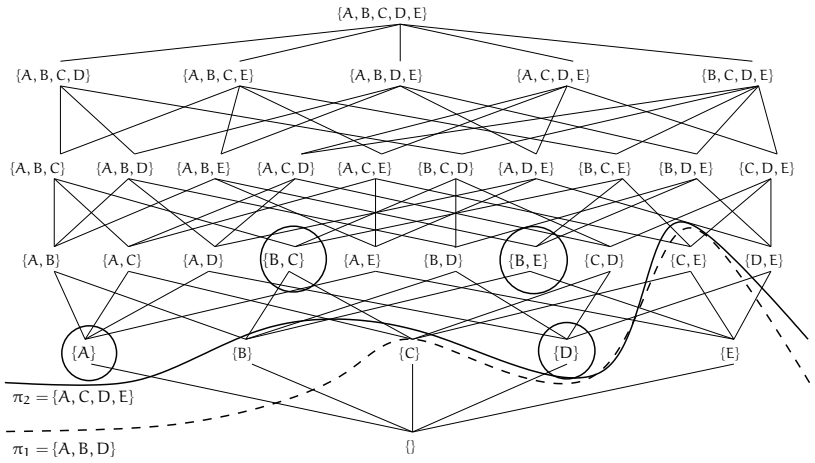


FIGURE 2.1: A lattice representing the diagnoses consistent with the two conflicts π_1 and π_2. The diagnoses are the sets above the line while the minimal diagnoses are circled.

empty diagnosis, i.e. $\{\{\}\}$, which means that all components are in the non-abnormal mode, while the top node is the diagnosis representing that every component are in the abnormal mode, i.e. $\{\{A, B, C, D, E\}\}$.

The following example illustrates the relation between conflicts, diagnoses, minimal diagnoses, and the lattice.

EXAMPLE 2.3: In Figure 2.1, a lattice representing diagnoses for a system with 5 components is shown. Two conflicts π_1 and π_2 have been introduced and the diagnoses should be found that are consistent with the conflicts.

The first conflict $\pi_1 = \{A, B, D\}$ states that there is a conflict that A, B, and D are in the non-abnormal mode. Therefore, since at least one of the components A, B, and D must be included in a diagnosis, the empty diagnosis, the single diagnoses $\{C\}$ and $\{E\}$, and the diagnosis $\{C, E\}$ are removed, because each of these diagnoses has an empty intersection with the conflict. In the lattice the dashed line represents this. The diagnoses below the line are inconsistent, i.e. they are in fact not diagnoses at all. The second conflict is $\pi_2 = \{A, C, D, E\}$ and the single diagnosis $\{B\}$ is inconsistent with this new conflict. Left are the diagnoses above the line in the figure where the minimal diagnoses are circled. ◇

Hitting sets

Theorem 2.2 is commonly used within diagnosis. Efficient algorithms for calculating the diagnoses given a set of diagnoses are therefore needed. The theorem can be seen as a special case of *hitting sets*, which also is denoted *vertex cover*.

DEFINITION 2.7 (Hitting set [Wot01]): *Let $\mathcal{F}$ be a set of sets. The set $S \subseteq \bigcup_{F \in \mathcal{F}} F$ is a hitting set for the set $\mathcal{F}$ if*

$$S \cap F \neq \emptyset$$

for all $F \in \mathcal{F}$.

DEFINITION 2.8 (Minimal hitting set [Wot01]): *A hitting set S' for the set $\mathcal{F}$ is a minimal hitting set if there is no proper subset*

$$S \subsetneq S'$$

where S is a hitting set for the set $\mathcal{F}$.

From the definitions can be seen that the diagnoses are the hitting sets for the set of conflicts. It is also the case that the *minimal diagnoses* are the *minimal hitting sets* for the set of *minimal conflicts*. Notice that Assumption 2.1 has to be true for these relationships to hold. In [dKMR92] is a proof for these relations given.

2.6 DIAGNOSTIC TESTS AND CONFLICTS

The evaluations of diagnostic tests are a common approach used to detect and isolate faults in a system. These tests might for example compare a sensor's value with some prediction of the sensor's value, and if these values fundamentally deviate from each other, it is concluded that some fault or faults are present in the system. In consistency based diagnosis, the result from the tests are stated as conflicts. The two examples below illustrate the relation between tests and conflicts.

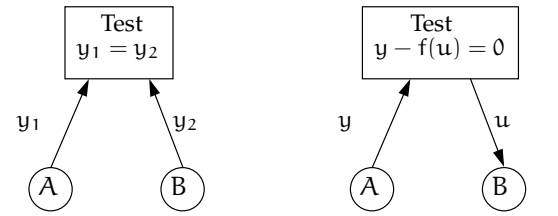
EXAMPLE 2.4: Consider a system including the two sensors A and B, which in some way measure the same temperature. If the values of sensor A and sensor B fundamentally deviate, then a conflict is that both these sensors are behaving non-abnormal. In set notation the conflict $\pi = \{\{A, B\}\}$. The example is schematically seen in Figure 2.2 where y_1 and y_2 are the values from the sensors. ◇

EXAMPLE 2.5: Consider now a system including a component A controlled by the actuator signal u and a sensor B with value y. A model

exists for the component and the sensor when they are in the non-abnormal modes

$$\neg AB(A) \rightarrow x = f(u)$$
$$\neg AB(B) \rightarrow y = x.$$

A test could be to check if $y - f(u)$ has a small value, i.e. if the model and the observations are consistent. If these are not consistent, then a conflict $\pi = \{A, B\}$ exists in the system. ◇



(a) Test used in Example. (b) Test used in Example.

FIGURE 2.2: Two different tests that show how components can be tested.

The design of tests demands expert domain knowledge and a good insight into diagnostic systems. See for example [PFC00, CP99, Nyb99, Fri01].

3

Distributed Diagnostic Systems

Consistency based diagnosis was introduced in Chapter 2. This type of diagnosis is extended in Section 3.1 with a theoretical framework that will be used later in the thesis. The framework aims at defining a system description that is useful considering distributed diagnosis. The framework is extended with diagnostic tests in Section 3.2.

Since distributed systems are considered, there is a possibility that a fault in one part of the system indirectly affects another part of the system. This could for example be thru faulty outputs. Fault propagation is discussed in Section 3.3. The relation between local and global diagnoses is discussed in Section 3.5.

The number of global diagnoses grows both with the number and the size of the local diagnoses; therefore it is sometimes necessary to only consider the more likely global diagnoses. One way to decide which diagnoses that are most likely is to use the cardinality of the diagnoses. In Section 3.7 it is discussed how the number of global diagnoses can be reduced and under which assumptions that the set of minimal cardinality diagnoses are the same as the set of most probable diagnoses.

The chapter is ended with a discussion of how the system description relates to an implemented system, notably a Scania heavy-duty vehicle.

3.1 SYSTEM DESCRIPTION IN A DISTRIBUTED ENVIRONMENT

The diagnostic system involves a set of agents $\mathcal{A}$ and a set of objects Θ which is the set of objects that can be diagnosed, by one or several agents, for abnormal behavior. Since an agent should be able to state the behavioral in one of its inputs, the components have to be extended to the set of objects.

DEFINITION 3.1 (Objects): *The set of objects is*

$$\Theta = \mathcal{C} \cup \mathsf{IN}$$

where $\mathcal{C}$ *is a set of components and* IN *is a set of inputs.*

An input can be in the abnormal or non-abnormal mode. The set of objects for agent A is $\mathcal{C}^A$, and these sets of objects do *not* need to be disjoint. If all of the sets where disjoint then each of the agents is an isolated system and there would be no need for a distributed diagnostic system.

The agents are connected to each other via a network, where an output signal in an agent is linked to input signals in one or several other agents. In a vehicle, the agents are the software programs that are implemented in the ECUs. The objects are the sensors, actuators, pipes, inputs, etc., which can be diagnosed by the agents. The output signals, which are values from sensors, to actuators, or from calculations, are made available to the other agents over the network. The set of all inputs and outputs are IN and OUT respectively. The model is described by the system description.

DEFINITION 3.2 (System description): *Given a system consisting of a set of agents* $\mathcal{A}$*. The system description is*

$$\mathsf{SD} = \bigcup_{A \in \mathcal{A}} \mathsf{SD}^A \cup \mathsf{STRU}$$

where SD^A *is the system description of agent* A *and* STRU *describes which outputs that are connected to which inputs.*

The definition of the agents' system descriptions and the structural description will follow in the upcoming sections.

3.1.1 Object Diagnoses

Since a local agent should be able to state a diagnosis that includes both components and inputs, the diagnoses defined in Chapter 2 can not be used directly.

DEFINITION 3.3 (Object diagnosis): *An object diagnosis* $D \subseteq \Theta$ *is a partial diagnosis*

$$D = \bigwedge_{\theta \in D} AB(\theta) \wedge \bigwedge_{c \in \Theta \setminus D} A(c).$$

The same notation will be used to represent both diagnoses, as defined in Chapter 2, and object diagnoses. Further, both of these will be denoted diagnoses when no misunderstanding is imminent.

3.1.2 Object conflicts

As with diagnoses, conflicts can also be extended to objects. The corresponding definition of object conflict is as follows.

DEFINITION 3.4 (Object conflict): *An object conflict* $\pi \subseteq \Theta$ *is a conflict*

$$D = \bigwedge_{\theta \in D} \neg AB(\theta).$$

An object conflict is a conflict defined over the set of objects.

3.1.3 Minimality operator

To extract the minimal sets of diagnoses, conflicts, or hitting sets, the minimal set representation operator is defined.

DEFINITION 3.5 (Minimal set representation): *Let* X *be a set of sets, then the minimal set representation of* X *is*

$$\min_S(X) = \{x | \nexists x', x' \subseteq x, x \in X, x' \in X\}.$$

If sets of diagnoses are considered, the equivalence of two sets of diagnoses states the relationship between minimal and non-minimal diagnoses.

DEFINITION 3.6 (Equivalence of sets of diagnoses): *Let* $\mathbb{D}^1$ *and* $\mathbb{D}^2$ *be two sets of diagnoses. If*

$$\min_S(\mathbb{D}^1) = \min_S(\mathbb{D}^2)$$

then the two sets of diagnoses are equivalent

$$\mathbb{D}^1 \simeq \mathbb{D}^2.$$

This equivalence relation will be used where two sets of diagnoses should be compared to each other.

EXAMPLE 3.1: Consider the sets of diagnoses $\{\{A, B, C\}, \{A, C\}\}$ and $\{A, C\}$. Both these sets are characterized by the minimal diagnosis

$\{A, C\}$ and therefore is

$$\{\{A, B, C\}, \{A, C\}\} \simeq \{A, C\}.$$

$\diamond$

3.1.4 System Description for an Agent

Each agent is described by a local system description. In this thesis, it is assumed that the model consists of equations, describing the non-abnormal behavior, and some assumptions, stating the demands for these equations to be correct. This means that no fault models are used.

DEFINITION 3.7 (Equation assumption): *Let* eq *be an equation, under the assumptions*

$$\mathrm{ass}(eq) \subseteq \Theta^A$$

the model behave as described by the equation.

Each assumption $c \in \mathrm{ass}(eq)$ is a representation in set notation of the mode assignment $\neg AB(c)$.

DEFINITION 3.8 (System description for agent A): *The system description for agent* A *is*

$$SD^A =< SE, CR >$$

where SE *is a set of equations and assumptions*

$$\mathrm{ass}(eq) \rightarrow eq$$

and CR *is a set of computation requirements*

$$E \rightarrow \mathtt{output}(A, i)$$

which defines which set of equations E *that is needed to compute a value for the given output. The set of equations* E *is a sub-set of the equations included in* SE.

The definition states that each SD^A is a tuple including equations with assumptions, and computation requirements for the outputs. Each computation requirement, in the set CR, describes which of the equations that are needed to calculate a specific output. In most cases, the computation requirements will be given directly by the equations.

However, this is not always the case, for example when there exist multiple ways to calculate an output.

EXAMPLE 3.2: If an agent is described by the following equations and assumptions

$$\begin{array}{lcll} ass(E_1) & \rightarrow & x & = sensor_1 \\ ass(E_2) & \rightarrow & x & = sensor_2 \\ ass(E_3) & \rightarrow & output(A,1) & = x \end{array}$$

then there are multiple ways to calculate the value of the output. For example from the first, the second, or from a mean value of the two sensor values. ◇

The computation requirements will be used later when looking at which objects whose abnormal behavior might affect the outputs. It is therefore essential that the computation requirements are given in the system description.

The following three-agent system will be used throughout the chapter to illustrate the different concepts introduced

EXAMPLE 3.3: The system is a simplification and modification of the intake system in a vehicle engine. The system includes five components, four inputs, three ECUs, and a network bus; see Figure 3.1. Each of the ECUs is modeled as one agent. The first agent deals with the intake, the second deals with the operation of the heat exchanger, while the third deals with the intake manifold of the engine.

The first agent A_1 uses a sensor to measure the environment temperature and makes this value available on the network. Agent A_2 measures the temperature in the cooler intake and the drop in temperature due to the heat exchanger. Finally, agent A_3 measures the temperature in the inlet manifold.

The component c_1 is the temperature sensor connected to A_1, c_2 is the intake pipes, c_3 is the heat exchanger, c_4 is a temperature sensor in the heat exchanger's cooler intake, and c_5 is the inlet manifold temperature sensor. The second agent has one input, while the third has three inputs. Together these components and inputs are the set of objects.

The system description for A_1 is

$$\begin{array}{rcl} SD^{A_1} =< & \{\{c_1, c_2\} \rightarrow & E_1^{A_1} : output(A_1, 1) = value(c_1)\}, \\ & \{\{E_1^{A_1}\} \rightarrow & output(A_1, 1)\} > \end{array}$$

The assumptions $\{c_1, c_2\}$ means that if $\neg AB(c_1) \wedge \neg AB(c_2)$ is true, then equation $E_1^{A_1}$ correctly models the system. It is not obvious that c_2 should be included in the assumptions. To decide this, expert knowledge of the system is needed. One reason to include c_2 would be if hot

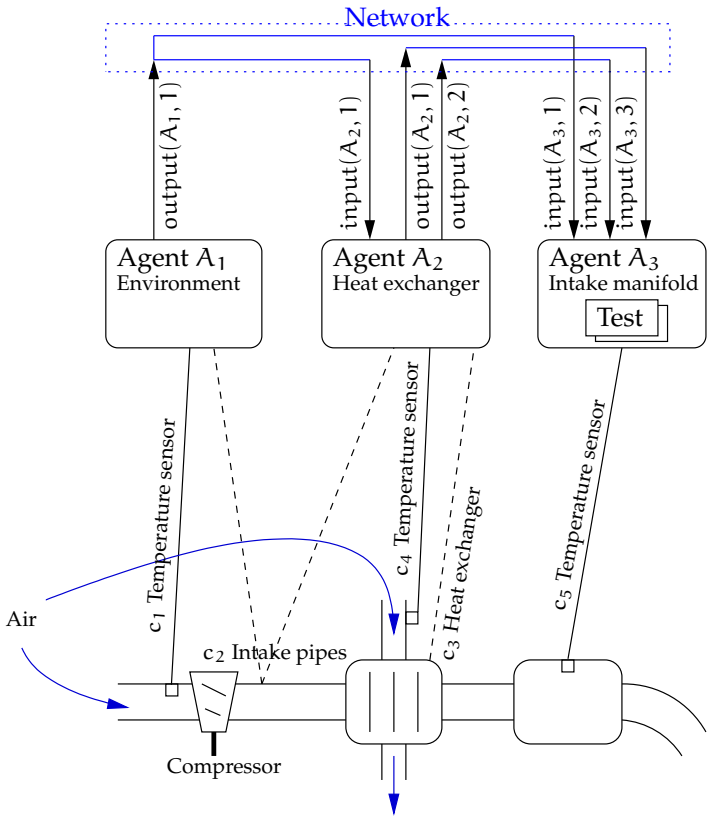


FIGURE 3.1: A system including three agents. The objects consist of five components and 4 inputs.

air could leak from the engine room into the intake pipes and thereby disturbing the temperature measuring.

Agent A_2 is modeled with four equations and it makes two outputs available on the network. The system description is

$$
\begin{aligned}
SD^{A_2} =< \quad & \{\{c_2, c_3, c_4\} \rightarrow \quad E_1^{A_2} : T_\Delta = \mu(T_s - value(c_4)), \\
& \{input(A_2, 1)\} \rightarrow \quad E_2^{A_2} : T_s = k \cdot input(A_2, 1), \\
& \emptyset \rightarrow \quad E_3^{A_2} : T_\Delta = output(A_2, 1), \\
& \{c_4\} \rightarrow \quad E_4^{A_2} : output(A_2, 2) = value(c_4)\}, \\
& \{\{E_1^{A_2}, E_2^{A_2}, E_3^{A_2}\} \rightarrow \quad output(A_2, 1), \\
& \{E_4^{A_2}\} \rightarrow \quad output(A_2, 2)\} >
\end{aligned}
$$

where μ and k are known constants.

Finally, the system description for agent A_3 is

$$\begin{aligned} SD^{A_3} =< \quad & \{\{c_5\} \rightarrow \quad E_1^{A_3} : T_{inlet} = value(c_5), \\ & \{input(A_3,1)\} \rightarrow \quad E_2^{A_3} : T_{air,1} = input(A_3,1), \\ & \{input(A_3,2)\} \rightarrow \quad E_3^{A_3} : T_{air,2} = input(A_3,2), \\ & \{input(A_3,3)\} \rightarrow \quad E_4^{A_3} : T_{\Delta} = input(A_3,3)\}, \\ & \emptyset > . \end{aligned}$$

The example will be continued later in the chapter. ◇

3.1.5 *The Structural Description*

The topology of the system, i.e. the connections between the inputs and outputs of the agents, is described by the structural description.

DEFINITION 3.9 (Structural description): *The structural description* STRU *is a set of equations*

$$input(A_k, i) = output(A_l, j)$$

where each equation defines a connection between the i*:th input in agent* A_k *and the* j*:th output in agent* A_l*.*

It will be shown later that it is sometimes useful to be able to extract which inputs that an output is connected to and vice versa. The following function is used to find the connections between inputs and outputs. Defined is also an extension that can be used to extract the agents that the inputs and outputs are connected to.

DEFINITION 3.10: *Given* $X \subseteq$ OUT *or* $X \subseteq$ IN*. The connection function is*

$$con(X) = \begin{cases} \{input(A,i) \mid `input(A,i) = x' \in STRU, x \in X\} & \text{if } X \subseteq OUT \\ \{output(A,i) \mid `x = output(A,i)' \in STRU, x \in X\} & \text{if } X \subseteq IN. \end{cases}$$

The connection to agent function is

$$con_A(X) = \begin{cases} \{A \mid `input(A,i)' \in con(X)\} & \text{if } X \subseteq OUT \\ \{A \mid `output(A,i)' \in con(X)\} & \text{if } X \subseteq IN. \end{cases}$$

The functions will also be used when the input is a scalar value and not a set. For example, $con(input(A_k, i)) = output(A_l, j)$. The structural description is exemplified in the following continuation of Example 3.3.

EXAMPLE 3.4: The structural description for the system is

$$\begin{aligned} STRU = \{ & input(A_2,1) = output(A_1,1), \\ & input(A_3,1) = output(A_1,1), \\ & input(A_3,2) = output(A_2,2), \\ & input(A_3,3) = output(A_2,1)\}. \end{aligned}$$

The connection function can be used to find the connections, for example

$$\mathrm{con}(\mathtt{input}(A_2, 1)) = \mathtt{output}(A_1, 1).$$

If the connection to agent function is used, then

$$\mathrm{con}_A(\mathtt{input}(A_2, 1)) = A_1.$$

$\diamond$

3.2 DIAGNOSTIC TESTS

Only pre-compiled tests are considered in this thesis, i.e. the tests are defined before the diagnostic system is employed. No local propagation, such as used in for example GDE [HCK92], is used.

DEFINITION 3.11 (Test): *Each agent* A *includes a set of tests* TEST^A*. Each test is a tuple*

$$t =< \mathsf{TC}, \pi >$$

where TC *is a test condition such that if*

$$\mathsf{TC}(\mathsf{OBS}) = \top$$

then a local conflict π *exist in the agent.*

The test condition can for example be designed as a residual that is compared with a threshold [Nyb99]. The set of tests in agent A is denoted TEST^A. If a set of equations E in the system description is tested for correctness by the test condition, then a reasonable conflict is

$$\pi = \bigcup_{eq \in E} \mathrm{ass}(eq).$$

EXAMPLE 3.5: Agent three is equipped with two pre-compiled tests

$$\mathsf{TEST}^{A_3} = \{< \mathsf{TC}_1, \pi_1 >, < \mathsf{TC}_2, \pi_2 >\}$$

where the test conditions are

$$|\mathsf{T}_{inlet} - k \cdot \mathsf{T}_{air,1} + \mathsf{T}_\Delta| > J$$
$$|\mathsf{T}_{inlet} - k \cdot \mathsf{T}_{air,2} + \mathsf{T}_\Delta| > J$$

for some threshold J. In the agent, the equations tested for correctness by the test conditions are

$$E_1 = \{E_1^{A_3}, E_2^{A_3}, E_4^{A_3}\}$$
$$E_2 = \{E_1^{A_3}, E_3^{A_3}, E_4^{A_3}\}.$$

This results in the conflicts

$$\pi_1 = \bigcup_{eq \in E_1} ass(eq) = \{c_5, input(A_3, 1), input(A_3, 3)\}$$
$$\pi_2 = \bigcup_{eq \in E_2} ass(eq) = \{c_5, input(A_3, 2), input(A_3, 3)\}.$$

Assume that both test conditions are true, then the set of local conflicts is $\Pi = \{\pi_1, \pi_2\}$. From these conflicts, the local minimal diagnoses

$$\mathbb{D}^{A_3} = \{\{c_5\}, \{input(A_3, 3)\}, \{input(A_3, 1), input(A_3, 2)\}\}$$

can be calculated. The set of diagnoses states that either are c_5, $input(A_3, 3)$, or both $input(A_3, 1)$ and $input(A_3, 2)$ behaving abnormal. If all diagnoses are considered, and not only the minimal diagnoses, then there are many more diagnoses, e.g. that both c_5 and $input(A_3, 3)$ are behaving abnormal at the same time. ◇

3.3 FAULT PROPAGATION

To calculate an output's value, it is sometimes needed to use several of the equations included in the system description. The correctness of an output does therefore not only depend on the assumption used in the equation defining the output, but also on assumptions in other equations that are needed to calculate the output.

DEFINITION 3.12 (Output assumption): *Let* $\sigma \in OUT^A$ *be an output in agent* A*, then the output assumption is*

$$ass(\sigma) = \bigcup_{eq \in E} ass(eq)$$

where E *is the equations in* $'E \rightarrow \sigma' \in CR^A$.

Notice that $ass(\sigma) \subseteq \Theta^A$. An output is said to be in the abnormal mode if any object included in its assumption is in the abnormal mode. Now a continuation of Example 3.3 is used to illustrate output assumptions.

EXAMPLE 3.6: Consider $output(A_2, 1)$ in agent A_2 whose defining equation is $E_3^{A_2}$. Included in CR is

$$\{E_1^{A_2}, E_2^{A_2}, E_3^{A_2}\} \rightarrow output(A_2, 1).$$

The output assumption is therefore

$$\begin{aligned} ass(output(A_2, 1)) &= \bigcup_{eq \in \{E_1^{A_2}, E_2^{A_2}, E_3^{A_2}\}} ass(eq) \\ &= \{c_2, c_3, c_4, input(A_2, 1)\}. \end{aligned}$$

If any of c_2, c_3, c_4, or $input(A_2, 1)$ is behaving abnormal, then this could result in the abnormal behavior of $output(A_2, 1)$. Since $input(A_3, 3)$ in agent A_3 is connected to this output, the abnormal behavior could be propagated from A_2 to A_3. ◇

If an input is included in an output assumption, then the output's behavior is dependent on the behavior of objects included in this input's assumption, i.e., the assumption for the output that the input is connected to. This means that an outputs behavior is indirectly dependent on the behavior of objects included in other agents.

It is sometimes useful to know exactly which components whose abnormal behavior could cause an output to behave abnormal. Therefore, the dependency function is defined. The function is used to find the components whose abnormal behavior could result in the abnormal behavior of an output. Defined is also a version for inputs.

DEFINITION 3.13 (Dependency): *Let $\sigma \in OUT$ be an output in agent A, then the dependency for σ is*

$$dep(\sigma) = ass(\sigma) \cap \mathcal{C} \cup \bigcup_{i \in ass(\sigma) \cap IN} dep(con(i)).$$

Let $i \in IN$ be an input, then the dependency for i is

$$dep(i) = dep(con(i)).$$

Since the function is defined implicit, the possibility of loops has to be considered in an implementation. The example is now extended to include output dependencies. A dependency $dep(i) \subseteq \mathcal{C}$.

EXAMPLE 3.7: The dependency of $input(A_3, 3)$ is

$$\begin{aligned} dep(input(A_3, 3)) &= dep(output(A_2, 1)) \\ &= ass(output(A_2, 1)) \cap \mathcal{C} \cup \bigcup_{i \in ass(output(A_2, 1)) \cap IN} dep(con(i)) \end{aligned}$$

where

$$\mathrm{ass}(\mathtt{output}(A_2, 1)) = \{c_2, c_3, c_4, \mathtt{input}(A_2, 1)\}.$$

This results in the dependency

$$\begin{aligned}\mathrm{dep}(\mathtt{input}(A_3, 3)) &= \{c_2, c_3, c_4\} \cup \mathrm{dep}(\mathrm{con}(\mathtt{input}(A_2, 1))) \\ &= \{c_2, c_3, c_4\} \cup \mathrm{dep}(\mathtt{output}(A_1, 1)) = \{c_1, c_2, c_3, c_4\}\end{aligned}$$

where $\mathrm{dep}(\mathtt{output}(A_1, 1) = c_1, c_2$ is taken directly from SD^{A_1}. The dependency will be used in the example in the following section. ◇

3.4 DISTRIBUTED DIAGNOSIS

In contrast to a centralized system where there exist one set of conflicts and one set of diagnoses, there can in a distributed environment exist several sets of conflicts and diagnoses. These sets of distributed conflicts and diagnoses are here denoted *local conflicts* and *local diagnoses*. The sets might be disjoint, i.e. independent, or not disjoint, i.e. include shared components.

A global diagnosis has the ability to state the mode of all components in the system. The global diagnoses can be formed directly from the local conflicts, where all conflicts are merged to a set of global conflicts and thereafter, the global diagnoses are generated from these global conflicts. An alternative is to first generate the local diagnoses from the local conflicts, then all local diagnoses are merged to obtain the set of global diagnoses. These approaches are schematically shown in Figure 3.2.

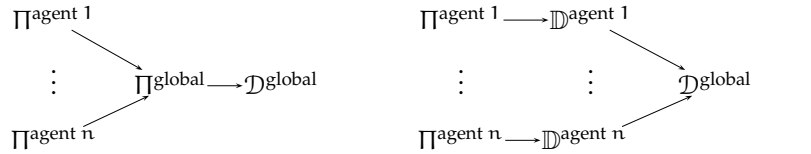
The computationally most expensive operation is the generation of local diagnoses and the merge of local diagnoses. Therefore, the first approach might be seen as a more centralized diagnostic system, while the second approach is more like a distributed diagnostic system.

When merging local diagnoses, the combinatorial explosion that this results in must be considered. In Chapter 6 a focused search will be presented that reduces the number of global diagnoses.

3.5 RELATION BETWEEN LOCAL AND GLOBAL DIAGNOSES

As described in the preceding section, distributed diagnosis aims at finding correct global diagnoses in a distributed environment.

The sets of local conflicts and diagnoses in agent A are denoted Π^A and $\mathbb{D}^A$ respectively. A global diagnosis is a diagnosis consistent



(a) Local conflict to global conflict to global diagnoses.

(b) Local conflict to local diagnoses to global diagnoses.

FIGURE 3.2: Two different approaches to generate global diagnoses.

with the complete system description. A global diagnosis might be represented as an object diagnosis. The $\varphi_{\mathcal{C}}$ operator can be used to transfer the global diagnosis to complete component representation.

The global diagnoses can be calculated from all the local conflicts as described in Theorem 2.2. A set of global diagnoses is denoted $\mathcal{D}$. The local diagnoses can be *merged* to form the global diagnoses. If the diagnoses are represented in boolean algebra, then the merge is a simple conjunction followed by the removal of non-minimal diagnoses. If set notation is used, then the *merging* of two sets of diagnoses is defined as follows.

DEFINITION 3.14 (Merge): *Let $\mathbb{D}^1$ and $\mathbb{D}^2$ be two sets of diagnoses, then a merge of these diagnoses is the set of minimal sets*

$$\mathbb{D}^1 \bowtie \mathbb{D}^2 = \min_{\mathrm{S}}(\{D^1 \cup D^2 \mid D^1 \in \mathbb{D}^1, D^2 \in \mathbb{D}^2\}).$$

The merge of local diagnoses is primary used to calculate global diagnoses from local diagnoses, which is an alternative to directly calculate the global diagnoses from the local conflicts. Theorem 3.1 states that the merge of local diagnoses gives the global diagnoses.

THEOREM 3.1 (From local diagnoses to global diagnoses): *For each $A \in \mathcal{A}$ let $\mathbb{D}^A$ be a set of local diagnoses consistent with the diagnoses Π^A. The minimal global diagnoses is*

$$\mathcal{D} = \bigbowtie_{A \in \mathcal{A}} \mathbb{D}^A.$$

Proof. A diagnosis $D \in \mathcal{D}$ is

$$D = \bigcup_{A \in \mathcal{A}} D^A$$

where $D^A \in \mathbb{D}^A$. For each conflict $\pi \in \bigcup_{A \in \mathcal{A}} \Pi^A$

$$D \cap \pi \neq \emptyset$$

since each D^A is a diagnosis, and from Theorem 2.2 follows that D is a global diagnosis. A merge only includes the minimal sets, and therefore the global diagnoses are minimal global diagnoses. □

EXAMPLE 3.8: Consider an example with two agents including the conflicts $\Pi^{A_1} = \{\{A, C\}, \{B, C\}\}$ and $\Pi^{A_2} = \{\{B, E\}\}$ respectively. The local minimal diagnoses consistent with these conflicts are $\mathbb{D}^{A_1} = \{\{A, B\}, \{C\}\}$ and $\mathbb{D}^{A_2} = \{\{B\}, \{E\}\}$. The set of merged local diagnoses is

$$\mathbb{D}^{A_1} \bowtie \mathbb{D}^{A_2} = \{\{A, B\}, \{B, C\}, \{C, E\}\}$$

The diagnosis $\{A, B\} \cup \{E\} = \{A, B, E\}$ is not minimal and is therefore not included in the merged set. Notice that every diagnosis is consistent with every conflict, i.e. every merged diagnosis is a global diagnosis. ◇

3.6 COMPLETE COMPONENT REPRESENTATION

When computing and merging object diagnoses, it is possible to replace the inputs in the object diagnoses with the corresponding dependencies. This can lead to both larger and smaller sets of object diagnoses depending on the structure of the system. The object diagnoses will be said to be *completely component represented* if $D \subseteq \mathcal{C}$, i.e. all *inputs* have been replaced with the corresponding dependencies.

The $\varphi_{\mathcal{C}}$ operator for a set of object diagnoses is used to translate object diagnoses to complete component representation.

DEFINITION 3.15 (Object diagnosis in complete component representation): *Let* D *be an object diagnosis, then the diagnosis' complete component representation is*

$$\varphi_{\mathcal{C}}(D) = \mathcal{P}$$

where $\mathcal{P}$ *is the minimal hitting sets for the set of sets*

$$\{\{c\} \mid c \in D \cap \mathcal{C}\} \cup \bigcup_{i \in D \cap IN} \{\{dep(i)\}\}.$$

For a set of object diagnoses $\mathbb{D}$*, the diagnoses' complete component representation is*

$$\varphi_{\mathcal{C}}(\mathbb{D}) = \min_S(\bigcup_{D \in \mathbb{D}} \varphi_{\mathcal{C}}(D)).$$

Notice that a diagnosis is $D = \{c_1, \ldots, c_n, dep(I_1), \ldots, dep(I_m)\}$, and the set of sets that a hitting set should be found for is

$$\{\{c_1\}, \ldots, \{c_n\}, \{dep(i_1)\}, \ldots, \{dep(i_m)\}\}.$$

For each set $P \subseteq \mathcal{P}$, $P \subseteq \mathcal{C}$. All hitting sets are represented by the set

$$\{D \cap \mathcal{C}\} \underset{i \in D \cap IN}{\bowtie} \{\{c\} \mid c \in dep(i)\}.$$

The minimal diagnoses representing this set are therefore the minimal diagnoses, which are represented by the minimal hitting sets. The two formulations are equivalent.

EXAMPLE 3.9: Consider the diagnosis

$$D = \{A, B, i\}$$

where $dep(i) = \{B, C\}$. The diagnosis complete component representation is the minimal hitting sets for the set of sets

$$\{\{A\}, \{B\}, \{B, C\}\}$$

which is the set of diagnoses

$$\varphi_{\mathcal{C}}(D) = \{\{A, B\}\}.$$

$\diamond$

The behavior of an input depends on the components included in the input's dependency. The following assumption, commonly known as exoneration, will be used in the thesis.

ASSUMPTION 3.1 (Exoneration): *Let σ be an output, then*

$$AB(i) \leftrightarrow \bigvee_{c \in dep(i)} AB(c).$$

Notice that the right implication always hold. The exoneration is the implication of the left arrow. This means that the abnormal behavior of a component included in an output's dependency results in the abnormal behavior of the output. This assumption will be discussed further in Section 3.7.5 page 50.

PROPOSITION 3.2: *Let $\mathbb{D}$ be a set of object diagnoses, then*

$$\varphi_{\mathcal{C}}(\mathbb{D})$$

is a set of kernel diagnoses.

Proof. An object diagnosis $D \in \mathbb{D}$ is an expression

$$D = \bigwedge_{\theta \in D} AB(c) \wedge \bigwedge_{c \in \Theta \setminus D} A(c)$$

such that $SD \cup OBS \cup \{D\}$ is consistent. Partitioning D into inputs and components, and using exoneration give

$$\begin{aligned} D &= \bigwedge_{c \in D \cap \mathcal{C}} AB(c) \wedge \bigwedge_{i \in D \cap IN} AB(i) \wedge \bigwedge_{c \in \Theta \setminus D} A(c) \\ &= \bigwedge_{c \in D \cap \mathcal{C}} AB(c) \wedge \bigwedge_{i \in D \cap IN} \bigvee_{c \in dep(i)} AB(c) \wedge \bigwedge_{c \in \Theta \setminus D} A(c). \end{aligned}$$

Extracting the disjunction give

$$D = \bigvee_{S \in \bar{S}} \bigwedge_{c \in S} AB(c) \wedge \bigwedge_{c \in \Theta \setminus D} A(c).$$

where $\bar{S}$ are the minimal hitting sets for the sets

$$\{\{c\} \mid c \in D \cap \mathcal{C}\} \cup \bigcup_{i \in D \cap IN} \{dep(i)\}.$$

Noticing that for $c \in S$, $AB(c) \wedge A(c) = AB(c)$, and that $D \cap \mathcal{C} \subseteq S$, give

$$D = \bigvee_{S \in \bar{S}} \bigwedge_{c \in S} AB(c) \wedge \bigwedge_{c \in \Theta \setminus S} A(c).$$

Now, each $S \subseteq \mathcal{C}$ and

$$SD \cup OBS \cup \{\bigwedge_{c \in S} AB(c) \wedge \bigwedge_{c \in \Theta \setminus S} A(c)\}$$

is consistent which means that each S is a partial diagnosis. The minimal hitting set represents the kernel diagnoses, and the kernel diagnoses characterizes all partial diagnoses, therefore is $\varphi_{\mathcal{C}}(D)$ a set of kernel diagnoses. From Definition 3.15 follows that $\varphi_{\mathcal{C}}(\mathbb{D})$ is a set of kernel diagnoses. □

The corresponding definition and proposition for conflicts are stated below.

DEFINITION 3.16 (Object conflict in complete component representation): *Let π be an object conflict, then the conflict's complete component representation is*

$$\varphi_{\mathcal{C}}(\pi) = \mathcal{C} \cap \pi \cup \bigcup_{i \in \pi \cap IN} dep(i).$$

For a set of object conflicts Π, the conflicts' complete component representation is the set

$$\varphi_{\mathcal{C}}(\Pi) = \min_S(\bigcup_{\pi \in \Pi} \varphi_{\mathcal{C}}(\pi)).$$

PROPOSITION 3.3: *Let* Π *be a set of object conflicts where* $\pi \subseteq \Theta$ *for each* $\pi \in \Pi$*, then*

$$\varphi_{\mathcal{C}}(\Pi)$$

is a set of minimal conflicts.

Proof. For $\pi \in \Pi$

$$SD \cup OBS \cup \{\bigwedge_{c \in \pi} \neg AB(c)\}$$

is inconsistent. Partition the conflict and using a negation of Assumption 3.1 give

$$\begin{aligned} &SD \cup OBS \cup \{\bigwedge_{c \in \pi \cap \mathcal{C}} \neg AB(c) \wedge \bigwedge_{i \in \pi \cap IN} \bigwedge_{c \in dep(i)} \neg AB(c)\} \\ =&SD \cup OBS \cup \{\bigwedge_{c \in \mathcal{C} \cap \pi \cup \bigcup_{i \in \pi \cap IN} dep(i)} \neg AB(c)\} \end{aligned}$$

where

$$\mathcal{C} \cap \pi \cup \bigcup_{i \in \pi \cap IN} dep(i) \subseteq \mathcal{C},$$

and by definition follows that

$$SD \cup OBS \cup \{\bigwedge_{c \in \varphi_{\mathcal{C}}(\pi)} \neg AB(c)\}$$

is inconsistent. The set $\varphi_{\mathcal{C}}(\pi)$ is therefore a conflict, and from the definition follows that the set $\varphi_{\mathcal{C}}(\Pi)$ is minimal conflicts. □

One of the cases where the operator is useful is when it should be verified that two sets of diagnoses are representing the same set of minimal diagnoses. The following relation will be used when verifying that two different sets of diagnoses are equal when all inputs have been replaced with the corresponding dependencies.

DEFINITION 3.17 ($\varphi_{\mathcal{C}}$ equivalence): *Let* X *and* Z *be two sets of diagnoses or conflicts. If*

$$\varphi_{\mathcal{C}}(X) = \varphi_{\mathcal{C}}(Z)$$

then the sets are $\varphi_{\mathcal{C}}$ *equivalent*

$$X \overset{\varphi_{\mathcal{C}}}{=} Z.$$

EXAMPLE 3.10: Consider the diagnoses in Example 3.5

$$\mathbb{D}^{A_3} = \{\{c_5\}, \{input(A_3, 3)\}, \{input(A_3, 1), input(A_3, 2)\}\}.$$

It would be interesting to exactly find which components whose abnormal behavior could give these diagnoses. The $\varphi_{\mathcal{C}}$ operator can be used for this.

To find $\varphi_{\mathcal{C}}(\mathbb{D}^{A_3})$, the dependency of all the inputs will first be calculated

$$\begin{aligned} dep(input(A_3), 1) &= \{c_1, c_2\} \\ dep(input(A_3), 2) &= \{c_2, c_4\} \\ dep(input(A_3), 3) &= \{c_1, c_2, c_3, c_4\}. \end{aligned}$$

The last dependency was calculated in Example 3.7. Now, the diagnoses can directly be transformed.

$$\begin{aligned} \varphi_{\mathcal{C}}(\{c_5\}) &= \{c_5\} \\ \varphi_{\mathcal{C}}(\{input(A_3, 3)\}) &= \{\{\}\} \uplus \{\{c\} \mid c \in dep(input(A_3, 3))\}\} \\ &= \{\{c_1\}, \{c_2\}, \{c_3\}, \{c_4\}\} \\ \varphi_{\mathcal{C}}(\{input(A_3, 1), input(A_3, 2)\}) &= \min_S(\{\{\}\} \uplus \{\{c_1\}, \{c_2\}\} \uplus \{\{c_2\}, \{c_4\}\}) \\ &= \{\{c_2\}, \{c_1, c_4\}\} \end{aligned}$$

which give the minimal diagnoses

$$\varphi_{\mathcal{C}}(\mathbb{D}^{A_3}) = \{\{c_1\}, \{c_2\}, \{c_3\}, \{c_4\}, \{c_5\}\}.$$

This states that at least one of the objects included in the diagnoses must behave abnormal.

It is also possible to first calculate the conflicts in flat form

$$\begin{aligned} \varphi_{\mathcal{C}}(\Pi) &= \{\{c_1, c_2, c_3, c_4, c_5\}\} \cup \{\{c_1, c_2, c_3, c_4, c_5\}\}) \\ &= \{\{c_1, c_2, c_3, c_4, c_5\}\}\}) \end{aligned}$$

The conflict can then be used to calculate the diagnoses stated above. ◇

3.7 PROBABILISTIC REASONING AND AVOIDANCE OF COMBINATORIAL EXPLOSION

When the diagnoses are calculated in a system, the number of possible diagnoses grows exponentially with both the number of local diagnoses and the size of the local diagnoses. This means that a combinatorial explosion is likely to occur when the number of objects in the

system grows. The problem is similar to the combinatorial problem of finding minimal diagnoses given a set of conflicts [dK92].

A first step to reduce the growth of the combinatorial explosion is to partition the global diagnoses into independent sub-sets of global diagnoses. How this can be done without losing too much of the global diagnoses properties will be discussed in Section 3.7.1.

To further reduce the growth of the combinatorial explosion, it is sometimes necessary to settle for the diagnoses that are the most interesting, and thereby reducing the number of diagnoses that have to be considered. A common approach to do this is to only consider the sub-set of diagnoses that *more probably* describes the current system behavior. One method to find the more probable diagnoses, is to use a-priori probabilities and probabilistic reasoning, such as used in [dK92]. There are however a number of problems with this approach which will be discussed in Section 3.7.2.

Due to the problems of deciding the probabilities for a diagnosis, a more coarse and simpler method can be used. Instead of using probabilities to prioritize between diagnoses, the size of the diagnoses can be used. This leads to the concept of *minimal cardinality* (mc) diagnoses, which will be discussed further in Section 3.7.3. The minimal cardinality concept is used together with the sub-sets of global diagnoses in Section 3.7.4.

The section is ended in Section 3.7.5 with a discussion about under which assumptions that the minimal cardinality diagnoses are the same as the most probable diagnoses.

3.7.1 Global Diagnoses represented as Module Diagnoses

For the diagnoses to be easily understandable for a technician, they should be as small as possible and being free of complex relations to other diagnoses. One such type of diagnoses that are free of complex relations are the global diagnoses, since each global diagnosis states a complete set of mode assignments. These might however become quite large due to the inclusion of all local diagnoses.

EXAMPLE 3.11: Consider two agents with local minimal diagnoses

$$\mathbb{D}^{\mathcal{A}_1} = \{\{A, B\}, \{C, F\}\}$$
$$\mathbb{D}^{\mathcal{A}_2} = \{\{B, C\}, \{C, D\}, \{F\}\}.$$

To repair the system, both sets of diagnoses have to be considered. This means that six combinations must be checked. However, the set of minimal global diagnoses is

$$\mathcal{D} = \{\{A, B, C\}, \{A, B, F\}, \{C, F\}\}$$

which is a simpler set of diagnoses to check. In a system with dozens of agents and many more diagnoses, the relations between the local diagnoses might become much more complex. ◇

One way to reduce the size of the global diagnoses is to represent them as a conjunction of smaller disjoint parts of diagnoses. Since these smaller parts are disjoint, the global diagnoses will simply be a merge of all these smaller parts, i.e. a simple relation would exist between the parts of diagnoses. When considering systems, such as a vehicle, the diagnoses will often be used for reparations. From a technician's point of view, the smaller diagnoses are more easily understandable than the complete set of global diagnoses.

One approach to extract these parts is to merge the local diagnoses from a sub-set of agents, into a sub-set of global diagnoses, so that each such sub-set is disjoint from the other. Such a set of agents is here denoted a *module*.

DEFINITION 3.18 (Module): *Let $\mathbb{D}^A$ be a set of diagnoses for agent A and partition the set of agents $\mathcal{A}$ into sub-sets $\bar{A}_i$. Each set $\bar{A}_i$ is said to be a module if for all sub-sets of agents*

$$\bar{A}_i \cap \bar{A}_j = \emptyset$$

and for all minimal diagnoses $D^{\bar{A}_i} \in \bowtie_{A \in \bar{A}_i} \mathbb{D}^A$, $D^{\bar{A}_j} \in \bowtie_{A \in \bar{A}_j} \mathbb{D}^A$

$$D^{\bar{A}_i} \cap D^{\bar{A}_j} = \emptyset.$$

The modules are defined with respect to some set of diagnoses $\mathbb{D}^A$. Therefore, the modules will be different when different sets of diagnoses are considered. This will for example be used in Chapter 6 to make the size of the modules smaller by only considering sub-sets of the local diagnoses.

For each module, the set of *module diagnoses* is defined.

DEFINITION 3.19 (Module diagnoses): *Let $\bar{A}$ be a module, and $\mathbb{D}^A$ a set of local diagnoses for agent A, then the set of module diagnoses is*

$$\mathcal{D}^{mod}_{\bar{A}} = \bowtie_{A \in \bar{A}} \mathbb{D}^A.$$

The global diagnoses can be calculated from all module diagnoses,

$$\mathcal{D} = \bowtie_{\bar{A}} \mathcal{D}^{mod}_{\bar{A}}.$$

This follows directly from Theorem 3.1. An example is used to illustrate the relation between module diagnoses and global diagnoses.

EXAMPLE 3.12: If $\mathbb{D}^{A_1} = \{\{A, B\}, \{C\}\}$, $\mathbb{D}^{A_2} = \{\{B, E\}, \{C\}\}$, and $\mathbb{D}^{A_3} = \{\{F\}\}$, then the modules are $\bar{\mathcal{A}}_1 = \{A_1, A_2\}$ and $\bar{\mathcal{A}}_2 = \{A_3\}$ since F is

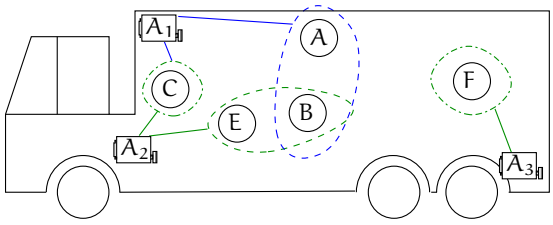


FIGURE 3.3: A schematic picture of a system consisting of three Agents. The agents includes local diagnoses (circled).

included in $\mathbb{D}^{A_3}$ but not in $\mathbb{D}^{A_1}$ or $\mathbb{D}^{A_3}$,. The sets of module diagnoses are

$$\mathcal{D}_1^{mod} = \min_S(\{\{A, B, E\}, \{A, B, C\}, \{C, B, E\}, \{C\}\}) = \{\{C\}, \{A, B, E\}\}$$
$$\mathcal{D}_2^{mod} = \{\{F\}\}.$$

The diagnoses are shown schematically in Figure 3.3 where the local diagnoses are represented by circles.

A merge of the module diagnoses results in

$$\mathcal{D}_1^{mod} \bowtie \mathcal{D}_2^{mod} = \{\{C, F\}, \{A, B, E, F\}\}$$

which also is the set of global diagnoses. ◇

3.7.2 *Probabilistic Reasoning*

There are a number of problems associated with the use of a-priori probabilities to prioritize between diagnoses. In a vehicle, such as a Scania truck, the probabilistic reasoning is difficult due to the difficulties in finding the a-priori probabilities. It is for a newly produced vehicle possible to find quite good estimates of the probabilities, however, after a couple of years, when for example some objects have been replaced or repaired, the estimates will probably be bad or completely faulty.

Due to the problem of deciding the probabilities for a diagnosis, a more coarse and simpler method can be used. Instead of using probabilities to prioritize between diagnoses, the size of the diagnoses can be used. This leads to the concept of minimal cardinality diagnoses.

3.7.3 *Minimal Cardinality Diagnoses*

The cardinality of a set is the number of elements in the set. The minimal cardinality diagnoses are a subset of all diagnoses.

DEFINITION 3.20 (Minimal cardinality diagnoses): *Let $\mathbb{D}$ be a set of diagnoses, then the set of minimal cardinality diagnoses is*

$$\mathbb{D}^{mc} = \{D \mid |D| = \min_{D \in \mathbb{D}} |D|, D \in \mathbb{D}\}.$$

The set of minimal cardinality local diagnoses for agent A is denoted $\mathbb{D}_A^{mc}$, and the set of minimal cardinality global diagnoses is denoted $\mathcal{D}^{mc}$. The minimal cardinality diagnoses are a sub-set of the minimal diagnoses, which also means that they do not represent all local diagnoses.

For a given set of diagnoses, the number of minimal cardinality diagnoses is often (but not always) less than the number of minimal diagnoses. These diagnoses can therefore be used to reduce the growth of the combinatorial explosion that arises when several sets of diagnoses should be merged together.

Merging all local diagnoses can according to Theorem 3.1, form the set of global diagnoses. Unfortunately, the relation between minimal cardinality local diagnoses and minimal cardinality diagnoses are not so simple. A merge of the minimal cardinality local diagnoses does *not* result in the minimal cardinality global diagnoses, i.e.

$$\mathcal{D}^{mc} \not\supseteq \bowtie_{A \in \mathcal{A}} \mathbb{D}_A^{mc}.$$

They are in general not even a sub-set of the merged diagnoses, i.e.

$$\mathcal{D}^{mc} \not\subset \bowtie_{A \in \mathcal{A}} \mathbb{D}_A^{mc}.$$

The following example proves this.

EXAMPLE 3.13: Consider Example 3.8 where $\mathbb{D}_{A_1}^{mc} = \{\{C\}\}$ and $\mathbb{D}_{A_2}^{mc} = \{\{B\},\{E\}\}$. The merge gives

$$\mathbb{D}_{A_1}^{mc} \bowtie \mathbb{D}_{A_2}^{mc} = \{\{C, B\},\{C, E\}\}$$

while

$$\mathcal{D}^{mc} = \{\{A, B\},\{B, C\},\{C, E\}\}$$

The minimal cardinality global diagnosis $\{A, B\}$ was not included in the merge of the minimal cardinality local diagnoses due to the common objects in the set of local diagnoses. ◇

3.7.4 Minimal Cardinality Module Diagnoses

In contrast to the case with merged minimal cardinality local diagnoses, a merge of the *minimal cardinality module diagnoses* (MCMD) does give the minimal cardinality global diagnoses. This is the main motivation for the use of the module diagnoses.

THEOREM 3.4: *Let $\mathcal{D}_i^{mod,mc}$ be the i:th set of* MCMDs*. The minimal cardinality global diagnoses*

$$\mathcal{D}^{mc} = \underset{i}{\bowtie} \mathcal{D}_i^{mod,mc}.$$

Proof. The set of global diagnoses is

$$\mathcal{D} = \underset{i}{\bowtie} \mathcal{D}_i^{mod}.$$

The minimal cardinality global diagnoses is the set

$$\begin{aligned}\mathcal{D}^{mc} &= \{D \mid |D| = \min_{D \in \mathcal{D}} |D|, D \in \mathcal{D}\} \\ &= \{D \mid |D| = \min_{D \in \mathcal{D}} |D|, \mathcal{D} = \underset{i}{\bowtie} \mathcal{D}_i^{mod}\}.\end{aligned}$$

The cardinality of the merge of two disjoint diagnoses is the sum of cardinality of the merged diagnoses. Since the different MCMDs are disjoint, the set

$$\begin{aligned}\mathcal{D}^{mc} &= \underset{i}{\bowtie}\{D \mid |D| = \min_{D \in \mathcal{D}_i^{mod}} |D|, D \in \mathcal{D}_i^{mod}\} \\ &= \underset{i}{\bowtie} \mathcal{D}_i^{mod,mc}.\end{aligned}$$

where the last equality follows from the definition of minimal cardinality module diagnoses. □

EXAMPLE 3.14: Consider Example 3.12 where the modules are $\bar{A}_1 = \{A_1, A_2\}$ and $\bar{A}_2 = \{A_3\}$. The sets of minimal cardinality module diagnoses are

$$\begin{aligned}\mathcal{D}_1^{mod,mc} &= \{\{C\}\} \\ \mathcal{D}_2^{mod,mc} &= \{\{F\}\}.\end{aligned}$$

A merge of the minimal cardinality module diagnoses results in

$$\mathcal{D}_1^{mod,mc} \bowtie \mathcal{D}_2^{mod,mc} = \{\{C, F\}\}$$

which also is the set of minimal cardinality global diagnoses. ◇

3.7.5 *When are the Minimal Cardinality Diagnoses the most Probable Diagnoses?*

For a diagnosis that only includes components, the set of minimal cardinality diagnoses are, under some specific assumptions, the same as the set of most probable diagnoses. However, this is not the case for general diagnoses. The minimal cardinality concept is therefore extended in the end of this section.

The motivation to the work presented in this section is to find out under which assumptions that the set of minimal cardinality diagnoses is the same as the set of most probable diagnosis.

The Probability for a Diagnosis

The a-priori probability for an object $c \in \Theta$ to be in the abnormal mode is

$$P(AB(c)).$$

The stochastic variable is that object c is in the abnormal mode, i.e. that object c is behaving abnormal. If all the objects are in the abnormal mode independently from each other, then the probability for the system to be in the mode where the objects $D \subseteq \Theta$ are in the abnormal mode is

$$P(D) = \prod_{c \in D} P(AB(c)) \prod_{c \in \Theta \setminus D} (1 - P(AB(c))). \tag{3.1}$$

Diagnoses Including Only Components

It will be assumed that the number of objects are limited in such a way that

$$|\Theta| \ll p^{-1}.$$

This means that the effect on the probability due to the number of objects is much lower than the effect of the removal or addition of an abnormal object.

LEMMA 3.5 (Minimal cardinality and most probable diagnoses): *Let* $\mathbb{D}$ *be a set of diagnoses, where* $D \subseteq \mathcal{C}$ *for each diagnosis* $D \in \mathbb{D}$. *If* $P(AB(c)) = p$ *for each* $c \in \mathcal{C}$ *where* $p < \frac{1}{2}$ *is some probability, then*

$$\mathbb{D}^{\textit{most probable}} = \mathbb{D}^{mc}$$

where

$$\mathbb{D}^{\textit{most probable}} = \{D \mid P(D) = \max_{D \in \mathbb{D}} P(D), D \in \mathbb{D}\}.$$

Proof. The probability for the most probable diagnoses is

$$\max_{D \in \mathbb{D}} P(D) = \max_{D \in \mathbb{D}} p^{|D|}(1-p)^{|\mathcal{C} \setminus D|} = p^{\min_{D \in \mathbb{D}} |D|}(1-p)^{|\Theta \setminus \min_{D \in \mathbb{D}} D|}$$

since $p < \frac{1}{2}$. The set of most probable diagnoses is

$$\begin{aligned}\mathbb{D}^{\text{most probable}} &= \{D \mid P(D) = p^{\min_{D \in \mathbb{D}} |D|}(1-p)^{|\Theta \setminus \min_{D \in \mathbb{D}} D|}, D \in \mathbb{D}\} \\ &= \{D \,\big|\, |D| = \min_{D \in \mathbb{D}} |D|, D \in \mathbb{D}\} = \mathbb{D}^{mc}\end{aligned}$$

where the last equality follows from the definition of minimal cardinality. □

A simple example will illustrate the result of the lemma.

EXAMPLE 3.15: Consider the global diagnoses $\{\{C, F\}, \{A, B, E, F\}\}$ from Example 3.12. Assume that all objects fail with the same small probability p. From Lemma 3.5 follows that the most probable diagnosis is

$$\mathbb{D}^{\text{most probable}} = \mathbb{D}^{mc} = \{\{C, F\}\}$$

and its probability

$$P(\{C, F\}) = \mathcal{O}(p^2).$$

◇

The Probability for a General Diagnosis

When diagnoses including both components and inputs are considered, the probabilities for the diagnoses are more difficult to approximate.

If only components are included in the diagnoses, then (3.1) holds for minimal diagnoses. However, when inputs are included in the diagnoses, it have to be considered that the probability for abnormal behavior of an input is

$$P(AB(i) \mid \bigvee_{c \in dep(i)} AB(c)) \leqslant 1$$

since it is not certain that an abnormal behavior in an object will lead to the abnormal behavior in a depending input. However, under Assumption 3.1 the probability is

$$P(AB(i) \mid \bigvee_{c \in dep(i)} AB(c)) = 1.$$

If the assumption is very far from true for some input, it should be considered if the components that break the assumption really should be included in the input's dependency. Consider now the following example.

EXAMPLE 3.16: The set of diagnoses in Example 3.5 is

$$\{\{c_5\}, \{input(A_3, 3)\}, \{input(A_3, 1), input(A_3, 2)\}\}.$$

The cardinality of the diagnoses are

$$\begin{aligned} |\{c_5\}| &= 1 \\ |\{input(A_3, 3)\}| &= 1 \\ |\{input(A_3, 1), input(A_3, 2)\}| &= 2. \end{aligned}$$

Under the assumptions that the components fails with the same small probability and exoneration, are the minimal cardinality diagnoses the same set as the most probable, as was the case in Lemma 3.5? Unfortunately not. The first diagnosis only includes component c_5 and its probability to fail is

$$P(\{c_5\}) = \mathcal{O}(p).$$

The second diagnosis is $\{\mathtt{input}(A_3, 3)\}$, whose probability

$$P(\{\mathtt{input}(A_3, 3)\}) \approx |\operatorname{dep}(\mathtt{input}(A_3, 3))| \cdot p = 4 \cdot p = \mathcal{O}(p).$$

The probability is approximately the number of dependent components times the probability p which is approximated by p.

The third diagnosis is $\{\mathtt{input}(A_3, 1), \mathtt{input}(A_3, 2)\}$ with cardinality two. A direct assumption would be that the probability for this diagnosis is lower than the other two, i.e. p^2. Expand the diagnosis with the input dependencies

$$\begin{aligned} &P(\{\operatorname{dep}(\mathtt{input}(A_3, 1)), \operatorname{dep}(\mathtt{input}(A_3, 2))\}) \\ &= P((AB(c_1) \vee AB(c_2)) \wedge (AB(c_2) \vee AB(c_4))) = \mathcal{O}(p) \end{aligned}$$

since c_2 is included in both dependencies. For simplicity, the non-abnormal components have been left out of the calculation. Notice that $P(\{\mathtt{input}(A_3, 1), \mathtt{input}(A_3, 2)\}) \neq \mathcal{O}(p^2)$, which might have been expected from a brief study of the diagnoses.

To conclude, only the two first diagnoses are included in the set of minimal cardinality diagnoses, but all three fails with approximately the same probability. ◇

From this example it is concluded that it is not sufficient to only use the cardinality to find the more likely diagnoses. To be more useful, the cardinality concept must be modified. This will be done in the next section where the following theorem is needed. Theorem 3.6 is a generalization of Lemma 3.5 to general diagnoses.

THEOREM 3.6 (Probability for a general diagnosis): *Let* $D \subseteq \Theta$ *be an object diagnosis. If* $P(AB(c)) = p$ *for each* $c \in \mathcal{C}$ *where* p *is some small probability, and Assumption 3.1 is true, then the probability for the diagnosis is*

$$P(D) = \mathcal{O}(p^{|S|})$$

where S *is a minimal hitting set for the set of sets*

$$\{\{c\} \mid c \in D \cap \mathcal{C}\} \cup \bigcup_{i \in D \cap IN} \{\operatorname{dep}(i)\}.$$

Proof. Following the proof in Proposition 3.2 give that the object diagnosis D is

$$D = \bigvee_{S \in \mathcal{S}} \bigwedge_{c \in S} AB(c) \wedge \bigwedge_{c \in \Theta \setminus S} A(c)$$

where S is the set of all hitting sets. The object diagnosis D is a set of diagnoses. From Lemma 3.5 follows that the most probable diagnoses are the diagnoses with all $A(c) = \neg AB(c)$, which are represented by the minimal hitting sets. The probability for each such diagnosis is $\mathcal{O}(p^{|S|})$ where S is a minimal hitting set, and p is small. The probability for the diagnosis is therefore

$$P(D) = \mathcal{O}(p^{|S|}).$$

□

An alternative way to describe the probability is that the probability is

$$P(D) = \mathcal{O}(p^{|D \cap \mathcal{C}| + |S'|})$$

where $S' = S \setminus (D \cap \mathcal{C})$ for some minimal hitting set S. The set S' is the set of components that are only included in the input dependencies. The term $|D \cap \mathcal{C}| + |S'|$ can be refereed to as the *extended* cardinality, compare with Lemma 3.5 where $|D|$ is the cardinality. The probability for the diagnoses in Example 3.16 is calculated in the following example.

EXAMPLE 3.17: The probability for the first conflict is simply

$$P(\{c_5\}) = \mathcal{O}(p^{1+0}) = \mathcal{O}(p).$$

The probability for the second diagnosis is

$$P(\{\mathtt{input}(A_3, 3)\}) = \mathcal{O}(p^{0+|S'|}).$$

Since $D \cap \mathcal{C} = \emptyset$, a minimal hitting set only includes one component, $|S'| = 1$ which results in the probability $P(\{\mathtt{input}(A_3, 3)\}) = \mathcal{O}(p)$.

The probability for the third diagnosis is

$$P(\{\mathtt{input}(A_3, 1), \mathtt{input}(A_3, 2)\}) = \mathcal{O}(p^{0+|S'|}).$$

The input dependencies are

$$\mathrm{dep}(\mathtt{input}(A_3, 2)) = \{c_2, c_4\}$$
$$\mathrm{dep}(\mathtt{input}(A_3, 1)) = \{c_1, c_2\}$$

and since $D \cap \mathcal{C} = \emptyset$, a minimal hitting set is $\{c_2\}$. This results in the probability

$$P(\{\mathtt{input}(A_3, 1), \mathtt{input}(A_3, 2)\}) = \mathcal{O}(p).$$

◇

3.7.6 Extended Cardinality

In the preceding section was the probability for a general diagnosis calculated. As stated in Theorem 3.6 and exemplified in Example 3.16, it is not possible to use the cardinality of a diagnosis to decide which diagnoses that are most probable. Therefore, some extension to the cardinality concept is needed. Inspired by Theorem 3.6, the *extended cardinality* is defined.

DEFINITION 3.21 (Extended cardinality): *Let* $D \subseteq \Theta$ *be an object diagnosis, then the extended cardinality of the diagnosis is*

$$\mathrm{ec}(D) = |S|$$

where S *is a minimal hitting set for the sets*

$$\{\{c\} \mid c \in D \cap \mathcal{C}\} \cup \bigcup_{i \in D \cap IN} \{\mathrm{dep}(i)\}.$$

From the extended cardinality follows the minimal extended cardinality diagnoses.

DEFINITION 3.22 (Minimal extended cardinality diagnoses): *Let* $\mathbb{D}$ *be a set of object diagnoses. The minimal extended cardinality diagnoses are the set*

$$\mathbb{D}^{\mathtt{mec}} = \{D \mid \mathrm{ec}(D) = \min_{D \in \mathbb{D}} \mathrm{ec}(D), D \in \mathbb{D}\}.$$

Now, given that Assumption 3.1 and the assumptions in Theorem 3.6 is true, the minimal extended cardinality diagnoses are the same as the most probable diagnoses. This motivates the use of the extended cardinality.

THEOREM 3.7: *Let* $\mathbb{D}$ *be a set of object diagnoses. If Assumption 3.1 and the assumptions in Theorem 3.6 is true, then the set of most probable diagnoses is the same as the set of minimal extended cardinality diagnoses,*

$$\mathbb{D}^{\textit{most probable}} = \mathbb{D}^{\mathtt{mec}}$$

where

$$\mathbb{D}^{\textit{most probable}} = \{D \mid P(D) = \max_{D \in \mathbb{D}} \mathcal{O}(P(D)), D \in \mathbb{D}\}.$$

Proof. From Theorem 3.6 follows that

$$P(D) = \mathcal{O}(p^{|S|})$$

where S is a minimal hitting set for the same set of sets as in the definition of extended cardinality. Using this in the most probable diagnoses

results in

$$\begin{aligned}\mathbb{D}^{\text{most probable}} &= \{D \mid \mathcal{O}(p^{|S(D)|}) = \max_{D\in\mathbb{D}} \mathcal{O}(p^{|S(D)|}), D \in \mathbb{D}\} \\ &= \{D \;\big|\; |S(D)| = \min_{D\in\mathbb{D}} |S(D)|, D \in \mathbb{D}\}\end{aligned}$$

where $S(D)$ is a minimal hitting set for diagnosis D, and it has been used that $p < \frac{1}{2}$. From the definition of extended cardinality and minimal extended cardinality follows that the most probable diagnoses is

$$\mathbb{D}^{\text{most probable}} = \{D \mid ec(D) = \min_{D\in\mathbb{D}} ec(D), D \in \mathbb{D}\} = \mathbb{D}^{mec}$$

where the last equality follows from the definition of minimal extended cardinality diagnoses. □

Now Example 3.16 is concluded with extended cardinalities.

EXAMPLE 3.18: The extended cardinalities of the diagnoses are

$$\begin{aligned}ec(\{c_5\}) &= 1 \\ ec(\{input(A_3, 3)\}) &= 1 \\ ec(\{input(A_3, 1), input(A_3, 2)\}) &= 1\end{aligned}$$

which follows from the calculations in Example 3.17. For comparison, the cardinalities of the diagnoses are

$$\begin{aligned}|\{c_5\}| &= 1 \\ |\{input(A_3, 3)\}| &= 1 \\ |\{input(A_3, 1), input(A_3, 2)\}| &= 2.\end{aligned}$$

The minimal extended cardinality diagnoses are the complete set of diagnoses while the minimal cardinality diagnoses are only the two first. The probabilities for the diagnoses are approximately

$$\begin{aligned}P(\{c_5\}) &= \mathcal{O}(p) \\ P(\{input(A_3, 3)\}) &= \mathcal{O}(p) \\ P(\{input(A_3, 1), input(A_3, 2)\}) &= \mathcal{O}(p)\end{aligned}$$

i.e. the probability is approximately the same for all three diagnoses.

To conclude, the set of minimal extended cardinality diagnoses is the same as the set of most probable diagnoses. ◇

3.8 SCANIA EQUIVALENCE

The system described in this chapter is a theoretical framework for distributed diagnostic systems. In a vehicle, the agents are the software programs that are implemented in the ECUs and the components

are the sensors, actuators, pipes, etc., which are monitored by the diagnostic system. The output signals are mostly values from sensors, actuator signals, or calculated values, that are made available to the other agents over the network. The output signals are collected by other agents and are primarily used for control and diagnosis.

In the current diagnostic system implemented in the Scania vehicles, there are two main types of tests implemented, *electric tests* and *plausibility tests*. Both of these are pre-compiled and are executed during the operation of the vehicle.

Electric tests are robust and simple to implement. The tests use simple models to supervise single components for correct behavior, and only with regards to static reference values for that component. They can usually not detect smaller errors, e.g. bias errors, unless the error offsets the signal such that it is outside the valid range.

Plausibility tests uses models, or relationships between components, to simultaneously supervise multiple components. They can detect smaller errors like bias errors, but is also more sensitive to noise. The conflicts are generated from tests that are local or, local with additional information gathered from other ECUs over the network. Both these types of tests falls nicely into the framework described above.

4

Extending Local Diagnoses

A local diagnosis states the status of local components and local inputs, while the status of the rest of the distributed system is unknown. An agent is not directly interested in the status of the rest of the system; it is however interested in information about components and inputs that are included in its own set of objects. To improve the local diagnoses, agents could share conflicts and diagnoses with each other.

The primary gain from this sharing of information is that an agent can state more complete diagnoses about its own objects, than it could do if it only used its own local conflicts. Thus, the local diagnoses can be extended as a result of the sharing of diagnostic information.

The sharing of information can primarily be done in two ways; either conflicts or diagnoses are shared, see Section 4.2 and 4.3 respectively. The extreme is to share all conflicts or diagnoses with all other agents, which however in most cases are both inefficient and unnecessary. Instead only the subset of conflicts or diagnoses that are of a direct interest to the specific agent should be transmitted.

The work presented in this and the next chapter was motivated by questions that aroused when considering which conflicts or diagnoses that should be transmitted, and when a conflict or diagnosis is transmitted, if it have to be transmitted in its completeness. If it would be possible to minimize the information that is transmitted then the size of the local diagnoses could be kept to a minimum, and the load on both the processing units and the network could be reduced.

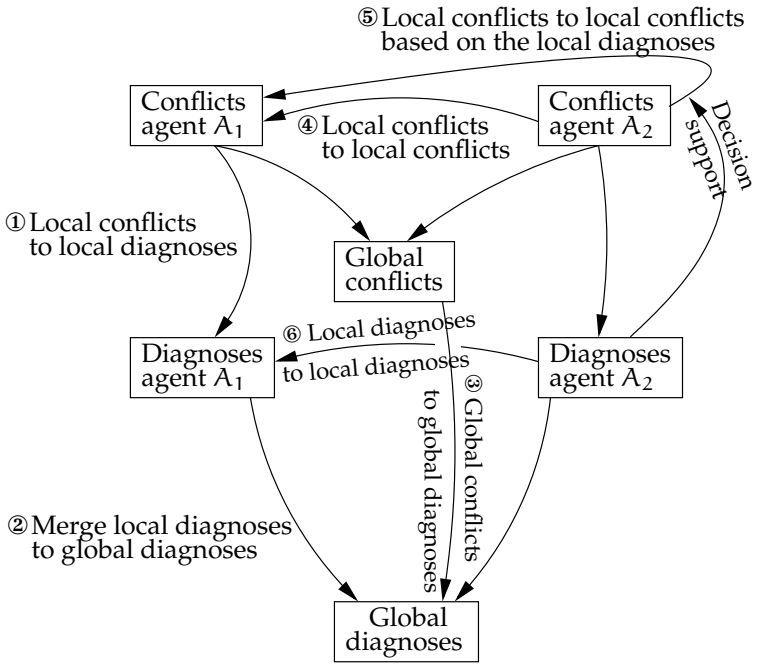


FIGURE 4.1: An overview of the different methods available for calculating and extending diagnoses.

4.1 DISTRIBUTION OF DIAGNOSTIC INFORMATION

Figure 4.1 shows the concepts that are involved in the generation and transmitting of both local and global conflicts and diagnoses. Two main concepts are shown in the figure; the calculation of diagnoses from conflicts and the distribution of conflicts and diagnoses from one agent to another. The two concepts are further described below.

4.1.1 From Conflicts to Diagnoses

The common approach to perform diagnosis is to first generate or detect conflicts and then computes the diagnoses based on the conflicts. In a distributed environment, where the goal is to compute the global diagnoses, this can be done in principally two different ways. From local conflicts generate local diagnoses, i.e. the standard approach within consistency based diagnosis, number ① in the figure. After this, merge the local diagnoses to obtain the global diagnoses

②. Alternatively, collect all the local conflicts to form a set of global conflicts, which can then be used to generate the global diagnoses ③. These approaches were briefly described in Section 3.4 where they were denoted *local conflicts to local diagnoses to global diagnoses* and *local conflicts to global conflicts to global diagnoses* respectively. Due to the combinatorial explosion it might be preferable to only calculate the MCMDs(minimal cardinality module diagnoses), and thereby reducing the number of diagnoses that have to be considered. An algorithm that finds all MCMDs is presented in Chapter 6.

4.1.2 Extending Local Conflicts and Local Diagnoses

As mentioned in the introduction to this chapter, two principally different ways can be found that results in extended local diagnoses.

First, it is possible to transfer a subset of local conflicts from one agent to another ④. This extends the local diagnoses in the receiving. Not all conflicts need to be transmitted. For example, only those conflicts that includes objects that is also included in the objects in agent A_1 could be transmitted. An alternative is to use the local diagnoses to decide which conflicts to transmit ⑤. These approaches will be denoted *local conflict to local conflict* and is discussed in more detail in Section 4.2

Second, a subset of local diagnoses can be transferred from one agent to another ⑥. The receiving agent can merge these diagnoses with its local diagnoses to obtain a more complete set of local diagnoses. This approach is denoted *local diagnoses to local diagnoses* and is discussed in Section 4.3.

4.2 SHARING LOCAL CONFLICTS

This section first discusses which conflicts that should be transmitted, and then how the size of the conflicts themselves can be reduced. But first, the following example is used to show how local conflicts can be shared between agents.

EXAMPLE 4.1: Consider a system consisting of two agents, with

$$\Pi^{A_1} = \{\{A, B\}\} \qquad \Pi^{A_2} = \{\{i, F, G\}\}$$
$$\mathrm{ass}(\sigma) = \{B\} \qquad STRU = \{i = \sigma\}$$

where $i = input(A_2, 1)$, $\sigma = output(A_1, 1)$, and Π^{A_i} the set of local conflicts in agent A_i, see Figure 4.2. The local diagnoses consistent with the conflicts are

$$\mathbb{D}^{A_1} = \{\{A\}, \{B\}\} \qquad \mathbb{D}^{A_2} = \{\{i\}, \{F\}, \{G\}\}.$$

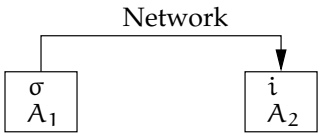


FIGURE 4.2: A two agent example.

Notice that A_1 has two diagnoses that are equally likely when the cardinality is considered. Agent A_2 detects that input i is possibly incorrect and therefore transmits this conflict to A_1. The transmitted conflict is

$$\pi^{tx} = \mathrm{con}(i) \cup \{F, G\} = \{\sigma, F, G\}$$

where tx is used to denote a transmitted conflict. The conflict is received, and the conflict's output is replaced with its dependency resulting in

$$\pi^{rx} = \mathrm{ass}(\sigma) \cup \{F, G\} = \{B, F, G\}$$

where rx is used to denote a received conflict. The received conflict results in the extended diagnoses for agent one

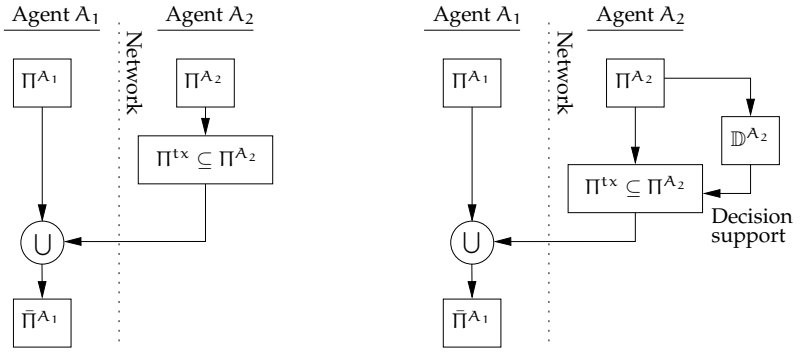
$$\bar{\mathbb{D}}^{A_1} = \{\{A, B\}, \{A, F\}, \{A, G\}, \{B\}, \{B, F\}, \{B, G\}\} \simeq \{\{A, F\}, \{A, G\}, \{B\}\}.$$

Notice that the diagnosis $\{B\}$ is now the most likely diagnosis. By sharing information between the diagnoses, agent A_1 obtained a more complete set of local diagnoses. ◇

4.2.1 *Different Approaches to Decide which Conflicts to Share Between the Agents*

The choice of which conflicts to distribute between agents can primarily be based on the local conflicts or on the local diagnoses. Figure 4.3 shows two schematic pictures illustrating how conflicts that should be transmitted are created from local conflicts.

In the second approach 4.3(b), the local diagnoses must be available, which however is not the case in the first approach 4.3(a). The gain from the second approach is that it is possible to only choose those conflicts that are more likely to include information that is of interest for the receiving agent. The downside is that the local diagnoses have to be calculated, and that the extended diagnoses will be less complete.



(a) Π^{tx} is transmitted to agent A_1 and joined into its conflicts to obtain the extended conflicts $\bar{\Pi}^{A_1}$.

(b) Based on the information in $\mathbb{D}^{A_2}$, a set of conflicts to be transmitted is chosen to be transmitted to the first agent.

FIGURE 4.3: Which conflicts to transmit from one agent to another can be decided in several different ways.

Based on Conflicts

To be able to decide which conflicts to transmit, it is necessary to estimate which conflicts that will give a desired increase in information in the receiving agent. A first step in choosing a set of conflicts is to only transmit the conflicts that include objects included in the receiving agent's objects.

A stricter limitation is to only transmit the conflicts that include inputs from the other agent. This is a natural idea since the exact knowledge about the objects might not be available.

The following example uses this stricter limit to decide which conflicts that should be transmitted to another agent.

EXAMPLE 4.2: Consider a system consisting two agents, where the first agent's conflicts are

$$\Pi = \{\{i_1, A\}, \{i_2, B\}, \{B, C\}\}$$

where both inputs are outputs from the same agent, i.e. $\mathrm{con}_A(i_1) = \mathrm{con}_A(i_2)$. It would at least be interesting for the second agent to receive the two first conflicts $\{i_1, A\}$ and $\{i_2, B\}$, since these includes information about outputs and thereby about the objects included in that agent. It might also be interesting to transmit the third conflict since component B is included in both the second and the third diagnosis. ◇

Based on the More Probable Diagnoses

To further limit the number of transmitted conflicts, it would be good to be able to quantify the amount of useful information that is included in the different conflicts. If the local diagnoses are available, this could be to only transmit conflicts that correspond to diagnoses that have a high probability to be correct.

One might for example only consider the conflicts that include inputs that are also included in the minimal cardinality local diagnoses. In general, a conflict $\pi \in \Pi^A$ should be transmitted if

$$(\pi \cap IN^A) \cap D \neq \emptyset.$$

for some diagnosis $D \in \mathbb{D}^A$ where

$$|D| \leqslant \min_{D \in \mathbb{D}^A} |D| + lim.$$

The limit lim could be used to decide which of the diagnoses that should be considered. If for example $lim = 0$ then only the minimal cardinality diagnoses are considered.

The following example uses the diagnoses to reduce the number of transmitted conflicts.

EXAMPLE 4.3: Consider Example 4.2 with the conflicts

$$\Pi = \{\{i_1, A\}, \{i_2, B\}, \{B, C\}\}.$$

The minimal diagnoses for the agent consistent with the conflicts are

$$\mathbb{D} \simeq \{\{i_1, B\}, \{A, B\}, \{i_1, i_2, C\}, \{A, i_2, C\}\}.$$

If the cardinality is used to decide which diagnoses that are most likely, then the first two are most likely. Based on this information, it seems more likely that i_1 is faulty than that i_2 is faulty; therefore to only transmit the first of the three conflicts seems reasonable. The number of transmitted conflicts has thereby been reduced compared to the number of transmitted conflicts in Example 4.2. ◇

A question that arises is if the same extended diagnoses will be found if the choice is based on the conflicts and when it is based on the diagnoses. The answer is that if the complete set of local diagnoses is considered, then the set of extended diagnoses will be the same. This follows from the fact that the conflicts completely characterize the diagnoses, which means that every object included in a conflict is included in a diagnosis. If only the more probable diagnoses is used, then the set of transmitted conflicts will be a sub-set of the conflicts transmitted if the choice was based on conflicts, and the set of extended diagnoses will therefore be reduced.

4.2.2 *Reducing the Size of Each Transmitted Conflict*

When transmitting a conflict from one agent to another, it is not certain that all the information in the conflict is of interest for the receiving agent. For example, the objects that are included in a conflict and whose behavior does not affect the receiving agent are not of any direct interest. Therefore, a reduction in the size of the extended diagnoses could be achieved if it was possible to only transmit the interesting part of the conflicts. Another gain is that the load on the network could be further reduced.

EXAMPLE 4.4: Consider once again Example 4.1. The extended local diagnoses in A_2 are

$$\bar{\mathbb{D}}^{A_1} \simeq \{\{A, F\}, \{A, G\}, \{B\}\}.$$

If the agent's objects are $\{A, B\}$, then it is from this agent's perspective quite unnecessary to have both F and G included in the diagnoses. The diagnoses $\{\{A, \Omega\}, \{B\}\}$ where Ω represents F or G would be sufficient. ◇

Is it possible to send only parts of the conflicts? In general, the answer is no, since only if the transmitted conflicts are super-sets of old conflicts then the diagnoses will be correct. This follows quite directly from the definitions of consistency based diagnosis. The following example illustrates why a transmitted conflict must be a super-set of an old conflict.

EXAMPLE 4.5: Continuation of example 4.1. Let the conflict transmitted from A_2 to A_1 be

$$\pi^{tx} = \{i\}$$

which is not a superset of any conflict in agent neither A_2 or A_1. The conflict is received and transformed to $\pi^{rx} = \{B\}$ in agent A_1. The extended diagnoses consistent with the old conflicts and this new conflict are

$$\bar{\mathbb{D}}^{A_1} = \{\{A, B\}, \{B\}\} \simeq \{\{B\}\}.$$

Is this a correct diagnosis? No, consider the merge of the extended local diagnoses which results in

$$\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2} = \{\{B\}\}$$

while the merge of the local diagnoses is the set

$$\mathbb{D}^{A_1} \bowtie \mathbb{D}^{A_2} = \{\{A, F\}, \{A, G\}, \{B\}\}$$

As expected, the merged global diagnoses are incorrect. ◇

Bundling the Non-Interesting Objects in the Transmitted Conflicts

The problem is that the complete conflict must be transmitted, while it is desired that its size is reduced. To solve this problem, it is here suggested that the non-interesting part of the conflict is bundled into one virtual object, denoted Ω. This object represents several other objects but has the same size as only one object.

When considering the system defined in Chapter 3, and basing the choice of which conflicts to transmit on the input-parts of the conflicts, then the relation between Ω and the rest of the objects can be described as follows. Partition the conflict π that should be transmitted from A_2 to A_1 into an input specific part $\mathcal{I}$ and the virtual object Ω^{A_2} so that

$$\pi^{tx} = \mathcal{I} \cup \{\Omega^{A_2}\}$$

where $\mathcal{I} = \pi \cap \mathrm{con}(\mathrm{OUT}^{A_1})$. The inputs $\mathrm{con}(\mathrm{OUT}^{A_1})$ specifies which inputs in IN^{A_2} that is of interest for A_1, and Ω^{A_2} represents the information that is not of interest for A_1.

The size of the transmitted conflict is thereby reduced to $|\mathcal{I}| + 1$, which often is smaller than the size of the original conflict. Notice that all Ω^A in the conflicts from agent A have been merged into one agent-specific Ω^A used in all conflicts transmitted from this agent. This results in diagnoses in the receiving agent that considers each agent as one abnormal object, i.e. there are not different objects of the agents that could be abnormal, but the agents as a single unit. An alternative that can be used to reduce the diagnoses even further is to merge all Ω^A into only one Ω. This results in diagnoses that consider all other agents as one abnormal object.

The following example exemplifies how the size of the receiving agent's diagnoses is reduced.

EXAMPLE 4.6: Consider Example 4.1 where the transmitted conflict with a virtual object is

$$\pi^{tx} = \{i\} \cup \{\Omega^{A_2}\}$$

which is received and transformed to

$$\pi^{rx} = \mathrm{ass}(\sigma) \cup \{\Omega^{A_2}\} = \{B, \Omega^{A_2}\}.$$

Resulting in the extended diagnoses

$$\bar{\mathbb{D}}^{A_1} = \{\{A, B\}, \{A, \Omega^{A_2}\}, \{B\}, \{B, \Omega^{A_2}\}\} \simeq \{\{A, \Omega^{A_2}\}, \{B\}\}.$$

The minimal diagnoses means that either is B abnormal *or* A and at least one object in agent A_2. The diagnosis $\{B\}$ is still the more likely diagnosis, while the number of diagnoses has been reduced from three to two. ◇

Merge of Diagnoses Including Virtual Objects

When the global diagnoses are calculated, the virtual objects have to be replaced with the original objects include in the conflicts. However, since the original information no longer is available, the virtual objects have to be replaced with some other set of objects. As long as this set is a super set of the objects that the virtual objects represent, the resulting diagnoses will be correct.

One such set is the objects supervised in the transmitting agent, i.e. the set Θ^A, another more pessimistic set is the set of all components, i.e. $\mathcal{C}$. Since these are super-sets of the original conflicts, they are also supersets of the objects represented by the virtual object, the resulting diagnoses are therefore correct.

When Ω is replaced in a diagnosis such as in Example 4.6, the following operator has to be used.

DEFINITION 4.1: *Let $\mathbb{D}$ be a set of diagnoses and Λ the set of virtual objects. If $\Omega \in \Lambda$ should be replaced with the objects $\Xi \subseteq \Theta$, then*

$$\Psi_\Omega(\mathbb{D}) = \min_S(\bigcup_{D\in\mathbb{D}} \Psi_\Omega(D))$$

where the diagnoses $\Psi_\Omega(D)$ are the minimal hitting sets for the set of sets

$$\{\{c\} \mid c \in D\backslash\Lambda\} \cup \bigcup_{\Omega\in\Lambda} \{\bar{\Omega}\}.$$

An alternative notation is that the sets $\Psi_\Omega(D)$ is the minimal diagnoses in the set of diagnoses

$$\{D\backslash\Lambda\} \cup \bigbowtie_{\Omega\in\Lambda} \{\{\omega\} \mid \omega \in \bar{\Omega}\}.$$

EXAMPLE 4.7: Consider the diagnosis

$$D = \{A, B, \Omega^{A_2}\}.$$

If Ω^{A_2} should be replaced by the set $\Xi^{A_2} = \{C, D\}$, then $\Psi_\Omega(D)$ are the minimal hitting sets for the sets

$$\{\{A\}, \{B\}, \{C, D\}.$$

which results in the sets

$$\Psi_\Omega(D) = \{\{A, B, C\}, \{A, B, D\}\}.$$

◇

Definition 4.1 is used in the following proposition.

PROPOSITION 4.1: *Let Π be a set of conflicts, and $\bar{\Pi}$ the set of corresponding conflicts where some components have been replaced with a virtual component in the set Λ. If $\bar{\mathbb{D}}$ is the set of diagnoses calculated from $\bar{\Pi}$, and $\mathbb{D}$ are calculated from Π, then*

$$\Psi_\Omega(\bar{\mathbb{D}}) \simeq \mathbb{D}.$$

Proof. Noticing that the Ψ_Ω operator resembles the $\varphi_{\mathcal{C}}$ operator if $D \backslash \Lambda$ is replaced with $D \cap \mathcal{C}$, $\Omega \in \Lambda$ is replaced with $i \in D \cap \mathbb{N}$, and $\bar{\Omega}$ with $\mathrm{dep}(i)$. The proof then follows from the proof in Proposition 3.2 Meaning that the virtual components are seen as inputs and theirs replacements are seen as the inputs dependencies. □

EXAMPLE 4.8: Consider Example 4.6. Merge the local diagnoses,

$$\varphi_{\mathcal{C}}(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2}) = \{\{B\}, \{A, F, \Omega^{A_2}\}, \{A, G, \Omega^{A_2}\}\}$$

which are the diagnoses calculated from the conflicts

$$\varphi_{\mathcal{C}}(\bar{\Pi}) = \{\{B, \Omega^{A_2}\}, \{A, B\}\} \cup \{\{B, F, G\}\}$$

If Ω^{A_2} is replaced with the pessimistic $\mathcal{C}$, then this give the diagnoses

$$\begin{aligned}\Psi_\Omega(\varphi_{\mathcal{C}}(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2})) \\ &= \min_S(\{\{B\}, \{A, F, A\}, \{A, F, B\}, \{A, F, F\}, \ldots, \{A, G, A\}, \ldots\}) \\ &= \{\{B\}, \{A, F\}, \{A, G\}\}\}.\end{aligned}$$

The set of minimal diagnoses calculated from the conflicts

$$\{\{B, F, G\}, \{A, B\}\} \cup \{\{i, F, G\}\}$$

is the set

$$\varphi_{\mathcal{C}}(\mathbb{D}^1 \bowtie \mathbb{D}^2) = \{\{B\}, \{A, F\}\{A, G\}\}$$

i.e. $\Psi_\Omega(\varphi_{\mathcal{C}}(\bar{\mathbb{D}}^1 \bowtie \mathbb{D}^2)) = \varphi_{\mathcal{C}}(\mathbb{D}^1 \bowtie \mathbb{D}^2)$ In contrast to the result in Example 4.5, the diagnoses are here correct.

Notice that in an implementation, the replacement of Ω^{A_2} would have skipped the middle step, i.e. $\{\{B\}, \{A, F\}, \{A, G\}\}$ would have been computed in a more direct way. ◇

4.3 SHARING LOCAL DIAGNOSES

Instead of transmitting local conflicts, it is possible to first calculate the local diagnoses and instead transmit a sub-set of the diagnoses.

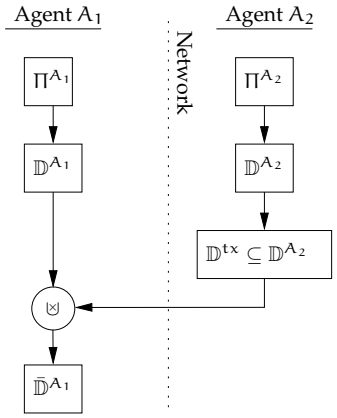


FIGURE 4.4: Local diagnoses to transmitted diagnoses.

A schematic picture of the local diagnoses to transmitted diagnoses approach is seen in Figure 4.4.

It might not be interesting for an agent to receive all diagnoses, for example might the diagnoses that have a low probability to be correct be uninteresting. It is therefore preferable to transmit the diagnoses that have a high probability to be correct. In a corresponding way as was done when conflicts was transmitted, only those diagnoses that include inputs from a second agent are transmitted to the corresponding agent.

When conflicts were transmitted, it was possible to choose the interesting conflicts and then directly transmit these to the receiver. This is however not the case when it comes to diagnoses, as will be exemplified in Example 4.9.

EXAMPLE 4.9: Consider the local diagnoses in Example 4.2,

$$\mathbb{D}^{A_1} = \{\{A\},\{B\}\} \qquad \mathbb{D}^{A_2} = \{\{i\},\{F\},\{G\}\}.$$

If the local diagnosis $\{i\}$ is transmitted to agent A_1 that transforms the diagnosis to $\{B\}$ and merges it with its own local diagnoses, then the extended local diagnoses become

$$\bar{\mathbb{D}}^{A_1} \simeq \{\{B\}\}.$$

This is however not a correct set of diagnoses. Notice for example that

$$\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2} = \{\{B\}\} \neq \mathbb{D}^{A_1} \bowtie \mathbb{D}^{A_2}.$$

$\diamond$

The reason why the simple approach in the example failed, is that it was not considered that there could exist other diagnoses in the transmitting agent, i.e. diagnoses that was not transmitted.

4.3.1 How to Transmit Local Diagnoses

When transmitting a sub-set of diagnoses, the information that these are only a sub-set of the diagnoses in the transmitting agent must be included when adding the diagnoses to the local diagnoses. This can be done with the addition of a supporting information, which is denoted S^A for the support from agent A. If a supporting information is included in a diagnosis, this means that there is, in addition to the objects in this specific diagnosis, something that is abnormal in the supporting agent A.

The following steps results in correct local extended diagnoses when transmitting diagnoses.

Let $\mathbb{D}^{A_1}$ be a set of diagnoses in A_1 and $\mathbb{D}^{A_2}$ a set of diagnoses in A_2. The set of transmitted diagnoses from A_2 to A_1 is $\bar{\mathbb{D}}^{tx}$ where each $D^{tx} \in \mathbb{D}^{tx}$ is based on a diagnosis $D \in \mathbb{D}^{A_2}$. Each transmitted diagnosis is transformed into

$$D^{tx} = con(\mathcal{J}) \cup (D \backslash \mathcal{J}) = \mathcal{P} \cup \bar{D}$$

where $\mathcal{J} = D \cap con(OUT^{A_1})$. The set of transmitted diagnoses is expanded into

$$\mathbb{D}^{tx} = \bar{\mathbb{D}}^{tx} \cup \{S^{A_2}\}$$

where S^{A_2} is the supporting information. The set is received and each received diagnosis is transformed into

$$D^{rx} = \underset{\sigma \in \mathcal{P}}{\bowtie} \{\{c\} \,|\, c \in ass(\sigma)\} \bowtie \bar{D}.$$

If the received diagnoses are merged with the local diagnoses, then a set of extended local diagnoses is formed where

$$\bar{\mathbb{D}}^{A_1} = \mathbb{D}^{A_1} \bowtie \mathbb{D}^{rx}$$

are the expanded local diagnoses.

If S is replaced in a set of diagnoses then the following operator has to be used.

DEFINITION 4.2: *Let $\mathbb{D}$ be a set of diagnoses and $\mathbb{S}$ the set of supporting informations. The set of diagnoses*

$$\Psi_S(\mathbb{D}) = min_S(\mathbb{D}\big|_{S=\emptyset, S \in \mathbb{S}}).$$

The definition states that the supporting informations should be replaced with the empty set. Definition 4.2 is used in the following proposition.

PROPOSITION 4.2: *A the merge of the extended local diagnoses results in the same set as the merge of the local diagnoses,*

$$\Psi_S(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2}) \overset{\varphi_e}{\simeq} \mathbb{D}^{A_1} \bowtie \mathbb{D}^{A_2}.$$

Proof. Consider a diagnosis that has received diagnoses including only one supporting information. The set of extended diagnoses is

$$\bar{\mathbb{D}}^{A_1} = \mathbb{D}^{A_1} \bowtie \mathbb{D}^{rx} = \mathbb{D}^{A_1} \bowtie \bar{\mathbb{D}}^{rx} \cup \mathbb{D}^{A_1} \bowtie \{S^{A_2}\}.$$

The merged set is therefore

$$\Psi_S(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2}) = \Psi_S(\mathbb{D}^{A_1} \bowtie \{S^{A_2}\} \bowtie \mathbb{D}^{A_2} \cup \mathbb{D}^{A_1} \bowtie \bar{\mathbb{D}}^{rx} \bowtie \mathbb{D}^{A_2}).$$

The set of received diagnoses are a subset of the set $\mathbb{D}^{A_2}$ if the inputs in the second set are replaced with the dependencies. The merged set in complete component representation is

$$\begin{aligned}\varphi_e(\Psi_S(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2})) &= \varphi_e(\min_S(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2})) \\ &\simeq \bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2})\end{aligned}$$

and by definition of $\overset{\varphi_e}{\simeq}$ follows that

$$\Psi_S(\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2}) \overset{\varphi_e}{\simeq} \mathbb{D}^{A_1} \bowtie \mathbb{D}^{A_2}.$$

The general case follows by iterative use of the proof for the system with two agents. □

Notice that S^{A_2} should be replaced with the empty set *after* the merge. If the replacement is done directly in the local extended diagnoses, then since the empty set is a minimal diagnosis, all other diagnoses are removed, which is incorrect. The following example exemplifies the steps outlined above.

EXAMPLE 4.10: Consider Example 4.1 where the diagnoses are

$$\mathbb{D}^{A_1} = \{\{A\}, \{B\}\} \qquad \mathbb{D}^{A_2} = \{\{i\}, \{F\}, \{G\}\}.$$

Agent A_2 has thus concluded that i is possibly incorrect and therefore transmit this diagnosis to A_1. Following the method outlined in Section 4.3.1, the transmitted diagnoses are

$$\mathbb{D}^{tx} = \{\{i\}, S^{A_2}\}$$

which is received and transformed into

$$\mathbb{D}^{rx} = \{\{B\}, S^{A_2}\}$$

A merge of the received and local diagnoses results in the extended diagnoses

$$\bar{\mathbb{D}}^{A_1} = \{\{A, B\}, \{A\} \cup S^{A_2}, \{B\}, \{B\} \cup S^{A_2}\} \simeq \{\{B\}, \{A\} \cup S^{A_2}\}.$$

It can be seen that the diagnosis $\{B\}$ is most likely when the cardinality is considered. The merge of the extended local diagnoses results in

$$\bar{\mathbb{D}}^{A_1} \bowtie \mathbb{D}^{A_2}|_{S=\emptyset} \overset{\varphi_{\mathcal{C}}}{\simeq} \{\{B\}, \{B, F\}, \{B, G\}, \{A, B\} \cup S^{A_2}, \{A, F\} \cup S^{A_2}, \{A, G\} \cup S^{A_2}\}|_{S=\emptyset} \simeq \{\{B\}, \{A, F\}, \{A, G\}\}$$

which is the represented by the same set of minimal diagnoses as in

$$\mathbb{D}^{A_1} \bowtie \mathbb{D}^{A_2} = \{\{B\}, \{A, F\}, \{A, G\}\}.$$

The merge results in the same set as the merge of the local diagnoses. ◇

4.3.2 *Reducing the Size of Transmitted Diagnoses*

When transmitting conflicts, a virtual object was added such that the size of the conflicts could be reduced. Both the size of the local extended diagnoses, and the load on the network could therefore be reduced. A corresponding approach could be used when transmitting diagnoses.

A diagnosis is partitioned such that

$$D^{tx} = \mathcal{I} \cup \Omega$$

where $\mathcal{I} = D \cap \mathrm{con}(\mathsf{OUT}^{A_1})$. When transmitting diagnoses, the virtual object can for example be replaced with the empty set, the original set or some super-set of objects. The global diagnoses will be correct in all three variations, since the supporting information is replaced with the empty set after the merge. Notice that this is not the same as when conflicts were transmitted, where the virtual object had to be replaced with a super-set of objects.

The downside when replacing Ω with the empty set is that information about the cardinality of the original diagnosis is lost. A direct extension would therefore be to include the cardinality information in the transmission.

5

Algorithms for Extending Local Diagnoses

In this chapter, some algorithms are presented that can be used to form local diagnoses according to the approaches described in Chapter 4. Firstly it is discussed how conflicts can be represented in different equivalent ways, see Section 5.1.

Two main approaches were discussed in the previous chapter, where the first was to extend the diagnoses thru the transmission of conflicts to other agents; this will be discussed in Section 5.2. The second was to transmit local diagnoses to the other agents, more about this in section 5.3. The chapter is concluded with a simulation study where these approaches have been evaluated.

5.1 Conflicts in Different Representations

The same conflict can be represented in several different ways. When transmitting conflicts or diagnoses, then the components that are inputs in the transmitting agent should be replaced with the corresponding output in the receiving agent. Further each of the outputs in the receiving agent should be replaced with its assumptions.

This means that a conflict π transmitted to agent A_1 should, when it is transmitted, be replaced with

$$\pi^{tx} = \mathrm{con}(\mathcal{I}) \cup \pi \backslash \mathcal{I}$$

where

$$\mathcal{I} = \pi \cap \mathrm{IN}^{A_1}.$$

The conflict is received and should then be replaced with

$$\pi^{rx} = \mathrm{ass}(\mathcal{P}) \cup \pi^{tx} \backslash \mathcal{P}$$

where $\mathcal{P} = \mathrm{con}(\mathcal{I})$. The conflict will then be a part of the objects in the receiving agent, i.e. $\pi^{rx} \subseteq \Theta^{A_1}$. The replacement of the inputs $\mathcal{I}$ with its connected outputs $\mathrm{con}(\mathcal{I})$ can be done either in the transmitting or in the receiving agent. The replacement of the outputs $\mathrm{con}(\mathcal{I})$ with its assumptions $\mathrm{ass}(\mathrm{con}(\mathcal{I}))$ is preferable done in the receiving agent. For diagnoses, an equivalent replacement should be performed.

5.2 EXTENDING DIAGNOSES THRU SHARING OF CONFLICTS

In Algorithm 1, the pseudo code for a general algorithm that extends diagnoses thru the transmission of conflicts is presented. In the algorithm, the $\mathrm{informed}$ variable includes information about which conflicts that has been transmitted to which agent. This is done to avoid re-transmission of the same information. In an implementation this should be used in a wider meaning where also minimal conflicts are considered, i.e. a set should not be transmitted if a set that is a sub-set of this conflict has already been transmitted.

The $\mathrm{InterestingInput}$ function defines which inputs that are of interest for other agents and is used to decide which conflicts that should be transmitted to the other agents. How to decide which inputs that are interesting was discussed in Section 4.2.1. If the set of interesting inputs is based on the set of local conflicts then

$$\mathrm{InterestingInput}(\Pi^A) \triangleq \mathrm{IN}^A \cap \bigcup_{\pi \in \Pi^A} \pi.$$

If it instead is based on the set of more probable local diagnoses then

$$\mathrm{InterestingInput}(\mathbb{D}^A) \triangleq \mathrm{IN}^A \cap \bigcup_{D \in \bar{\mathbb{D}}^A} D$$

Algorithm 1 Extended diagnoses for agent A.

Require: The conflicts Π^A in agent $A \in \mathcal{A}$.
Ensure: The extended diagnoses $\mathbb{D}^A$ in agent A.

1: $\text{informed} := \emptyset$
2: **repeat** [*Transmit and receive repetitious.*]
3: **for all** conflicts $\pi^{rx} = \mathcal{P} \cup \Omega^{A_t}$ received in agent A **do**
4: $\Pi^A := \min_S(\Pi^A \cup \{\bigcup_{\sigma \in \mathcal{P}} \text{ass}(\sigma) \cup \{\Omega^{A_t}\}\})$
[*Replace the outputs with their assumptions.*]
5: **end for**
6: If Π^A has been expanded then update $\mathbb{D}^A$ to be consistent with Π^A
7: $P := \text{InterestingInput}(\Pi^A)$
[*Based on which information should conflicts be transmitted?*]
8: $T = \{(\pi, \text{con}_A(i)) \mid i \in P, \pi \in \Pi^A\}$
[*Choose appropriate conflicts based on* P.]
9: $T := T \setminus \text{informed}$
10: **for all** $(\pi, A_r) \in T$ **do**
11: transmit $\pi^{tx} := (\text{con}(\pi \cap IN^A) \cap OUT^{A_r}) \cup \{\Omega^A\}$ to agent A_r
[*Replace inputs with the connected outputs.*]
12: **end for**
13: $\text{informed} := \text{informed} \cup T$
14: **until** The coordinator agent sends a break
[*End when no agent have transmit conflicts.*]
15: $\mathbb{D}^A :=$ diagnoses consistent with Π^A [*Find the extended diagnoses.*]

where $\bar{\mathbb{D}}^A$ is a subset of $\mathbb{D}^A$,

$$\bar{\mathbb{D}}^A = \{D \mid D \in \mathbb{D}^A, \mathtt{card}(D) \leqslant \min_{D \in \mathbb{D}^A}(\mathtt{card}(D)) + \mathtt{lim}\}$$

where $\mathtt{card}(D)$ is the cardinality of D. The limit $\mathtt{lim}$ is used to decide which local diagnoses that should be considered when deciding the interesting inputs. The algorithm uses the cardinality to focus the diagnoses. When the cardinality of Ω^A is calculated, then Ω^A will be seen as 1 component. An alternative way to calculate the cardinality of the diagnoses would be to use the extended cardinality, which was defined in Section 3.7.5.

The correctness of the result from the algorithm follows from Proposition 4.1. Notably is

$$\underset{A \in \mathcal{A}}{\bowtie} \mathbb{D}^A = \underset{A \in \mathcal{A}}{\bowtie} \bar{\mathbb{D}}^A$$

where $\mathbb{D}$ is a set of local diagnoses, and $\bar{\mathbb{D}}$ is sets of extended local diagnoses.

The algorithm terminates when all $T = \emptyset$. The set T is limited since both the included π and $\mathcal{A}$ are limited. The set informed increases when $T \neq \emptyset$ and therefore, at some time, $T = \emptyset$ in all agents, i.e. the algorithm terminates.

5.3 EXTENDING DIAGNOSES THRU SHARING OF DIAGNOSES

In Algorithm 2, a general algorithm that extends the local diagnoses thru transmitted diagnoses is presented. The virtual component Ω is defined according in Section 4.3.2, and once again, the informed variable is included to avoid re-transmission of the same information.

In the algorithm, the function UpdateExtendedDiagnoses is used to update the local diagnoses with the received diagnoses. This function is important and will therefore be discussed in deeper detail in Section 5.2.

If all different S^A are replaced with the empty set, the different Ω^A by sets of components as described in Section 4.3.2, and UpdateExtendedDiagnoses is correct, then Algorithm 2 is correct and terminates. The correctness of the algorithm is straightforward if function UpdateExtendedDiagnoses is correct. Termination is proved in the same way as termination is proved for Algorithm 1.

5.3.1 Update Extended Diagnoses

If a maximum of one set of diagnoses is transmitted from each agent, then a simple merge can be used as function UpdateExtendedDiagnoses, i.e.

$$\text{UpdateExtendedDiagnoses}(\mathbb{D}^A, \mathbb{D}^{rx}) := \mathbb{D}^A \bowtie \mathbb{D}^{rx}$$

where $\mathbb{D}^{rx}$ is the set of received diagnoses. However, if several sets of transmitted diagnoses can be received then this simple merge will become very inefficient. The following example illustrates the inefficiency.

EXAMPLE 5.1: Consider an agent with $\mathbb{D}^{A_2} = \{\{A\}, \{B\}\}$, if the first diagnosis is transmitted to an agent with an empty set of diagnoses $\mathbb{D}^{A_1} = \{\{\}\}$, then the extended set of diagnoses is $\mathbb{D}^{A_1} := \mathbb{D}^{A_1} \bowtie \{\{A\}, S^{A_2}\} = \{\{A\}, S^{A_2}\}$. If after some time it is found that the second diagnosis should be transmitted, then

$$\mathbb{D}^{A_1} := \mathbb{D}^{A_1} \bowtie \{\{B\}\} = \{\{A, B\}, \{B\} \cup S^{A_2}\}.$$

This shows that correct diagnoses was calculated but it was done in a quite inefficient way. Notice for example that the non-minimal

Algorithm 2 Extended diagnoses thru transmitted diagnoses

Require: The diagnoses $\mathbb{D}^A$ for agent $A \in \mathcal{A}$.
Ensure: The extended diagnoses $\bar{\mathbb{D}}^A$.

1: $\text{informed} := \emptyset$
2: **repeat** [*Transmit and receive repetitious.*]
3: **for all** diagnoses $D^{rx} \in \mathbb{D}^{rx}$ received from agent A_i where $D^{rx} = \mathcal{P} \cup \Omega^{A_t}$ **do**
4: $\mathbb{D}^{tx} := \{\bigcup_{\sigma \in \mathcal{P}} \text{ass}(\sigma) \cup \{\Omega^{A_t}\} \mid D \in \mathbb{D}^{A_i}\}$ [*Replace the outputs with its assumptions.*]
5: $\mathbb{D}^A := \text{UpdateExtendedDiagnoses}(\mathbb{D}^A, \mathbb{D}^{rx})$ [*Update the diagnoses with the transmitted diagnoses.*]
6: **end for**
7: $P := \text{InterestingInput}(\mathbb{D}^A)$ [*On what information should diagnoses be transmitted?*]
8: $T = \{(D, \text{con}_A(i)) \mid i \in P, D \in \mathbb{D}^A\}$ [*Choose appropriate diagnoses based on* P.]
9: $T := T \setminus \text{informed}$
10: **for all** $A_t \in \mathcal{A}$ **do**
11: $\mathbb{D}^{tx} := \{(\text{con}(D \cap IN^A) \cap OUT^{A_t}) \cup \Omega^A \mid (D, A_t) \in T\}$ [*Replace inputs with the connected outputs.*]
12: transmit $\mathbb{D}^{tx} \cup S^A$ to A_t if it is the first set transmitted to A_t, otherwise only transmit $\mathbb{D}^{tx}$
13: **end for**
14: $\text{informed} := \text{informed} \cup T$
15: **until** All agents have $T = \emptyset$. [*End when no agent have transmitted new diagnoses.*]

diagnosis $\{A, B\}$ is unnecessary calculated. The problem grows exponentially with the number of diagnoses. ◇

The problem is that the previously transmitted diagnoses are merged with the diagnoses that are transmitted in the upcoming transmissions. When accepting multiple sets of transmitted diagnoses from the same agent, a more efficient approach would be to remember which diagnoses that have already been received. An efficient approach for the above example would be to do as shown in the following example.

EXAMPLE 5.2: Store original diagnoses $T_0 := \mathbb{D}^{A_1}$. The first update gives $\mathbb{D}^{A_1} = T_0 \bowtie \{\{A\}, S^{A_2}\} = \{\{A\}, S^{A_2}\}$. The second update first calculates the transmitted diagnoses $M = T_0 \bowtie \{\{B\}\}$ and add these to the first update,

$$\mathbb{D}^{A_1} = \mathbb{D}^{A_1} \cup M = \{\{A\}, \{B\}, S^{A_2}\}.$$

This gives correct global diagnoses in a more efficient way. ◇

Algorithm 3 Update extended diagnoses.

Require: The i:th set of transmitted diagnoses from A_l, $\mathbb{D}_i^l$. The starting diagnoses $\mathbb{D}^A$ in the updating agent. Starting set $T_0 := \mathbb{D}^A$.
Ensure: The updated extended diagnoses $\mathbb{D}^A$.

[*Given that diagnoses have been received from* $A_1, \ldots, A_k$.]

1: **if** An agent denoted A_{k+1} have transmitted its first set of diagnoses, $\mathbb{D}_1^{k+1}$ **then**
2: $T_{k+1} := T_k \bowtie \mathbb{D}_1^{k+1}$ [*Merge all previous diagnoses with the new.*]
3: $\mathbb{D}^{k+1} := \mathbb{D}_1^{k+1}$ [*Store for later updates.*]
4: **end if**
5: **if** An old agent A_l have transmitted a set of diagnoses $\mathbb{D}_i^l$ **then**
6: $M_l := T_{l-1} \bowtie \mathbb{D}_i^l$ [*Merge all previous with the new.*]
7: $T_l := T_l \cup M_l$
[*Update the merged diagnoses up to this agent.*]
8: **for all** $A_{l+1}, \ldots, A_j, \ldots, A_k$ **do** [*Update the diagnoses received from later agents.*]
9: $M_j := M_{j-1} \bowtie \mathbb{D}^j$
10: $T_j := T_j \cup M_j$
11: **end for**
12: $\mathbb{D}^l := \mathbb{D}^l \cup \mathbb{D}_i^l$ [*Store for later use.*]
13: **end if**
14: $\mathbb{D}^A := T_k$ or T_{k+1} if it exists, i.e. the last T. [*The extended diagnoses.*]

The inefficiency becomes even more tedious when multiple agents send multiple sets of diagnoses.

In Algorithm 3, function UpdateExtendedDiagnoses is implemented in the efficient way that was exemplified in Example 5.2. When receiving a set of diagnoses, two different cases can occur. If an agent is transmitting its first set of diagnoses, then all the old updated diagnoses are merged with this transmitted set, denoted T_{k+1} in the algorithm. The received diagnoses are also stored for later use in $\mathbb{D}^{k+1}$.

Otherwise, an agent that has already transmitted at least one set of diagnoses has transmitted a new set. If it was the last agent that transmitted diagnoses then the new set of diagnoses are simply calculated from the preceding set T and the transmitted diagnoses. However, if it was one of the earlier agents, then the rest of the agent's calculated diagnoses T must be updated. This is performed in the loop.

LEMMA 5.1: *(Correctness of Algorithm 3) Given a set of diagnoses* $\mathbb{D}^A$ *in agent* A*, and that a set of agents have transmitted one or several sets of diagnoses. Then the result of Algorithm 3 is a set of diagnoses consistent with* $\mathbb{D}^A$ *and all received diagnoses.*

Proof. The updated $\mathbb{D}^A$ is correct if it can be shown that

$$T_k = \mathbb{D}^A \overset{k}{\underset{i=1}{\bowtie}} \mathbb{D}_{rx}^{A_i}$$

where $\mathbb{D}^A$ is the original set of diagnoses in A, and $\mathbb{D}_{rx}^{A_i}$ is the union of all received sets from A_i.

Firstly, if no diagnoses have been received from any agent, then $T_o = \mathbb{D}^A$. This is immediately a correct set of diagnoses. In previous runs, several sets of diagnoses could have been received from multiple agents. Let k be the number of agents that have sent diagnoses. If the old set $\mathbb{D}^A$ is assumed correct, then this set is consistent with all the original non-extended local diagnoses and with all transmitted diagnoses. The correctness of the last update will now be shown by induction to be correct. Let A_l be the last agent that transmitted a set, and let $\hat{\mathbb{D}}^l$ denote this set.

Assume that the all old $T_j = \mathbb{D}^A \bowtie \overset{j}{\underset{i=1}{\bowtie}} \mathbb{D}_{rx}^{A_i}$ for all j for 1 to k, i.e. they are correct. If agent A_l has not transmitted any diagnoses to this agent before, then

$$T_{k+1} = T_k \bowtie \mathbb{D}_1^{k+1} = \mathbb{D}^A \overset{k+1}{\underset{i=1}{\bowtie}} \mathbb{D}_{rx}^{A_i}$$

which is a correct set of diagnoses. Otherwise, the agent has transmitted diagnoses before. It should be shown that the updated T_k, which here is denoted $\hat{T}_k$, is

$$\hat{T}_k = \mathbb{D}^A \overset{l-1}{\underset{i=1}{\bowtie}} \mathbb{D}_{rx}^{A_i} \bowtie \left(\mathbb{D}_{rx}^{A_l} \cup \hat{\mathbb{D}}_{rx}^{A_l} \right) \overset{k}{\underset{i=l+1}{\bowtie}} \mathbb{D}_{rx}^{A_i}.$$

It can be rewritten as

$$\hat{T}_k = T_k \cup \mathbb{D}^A \overset{l-1}{\underset{i=1}{\bowtie}} \mathbb{D}_{rx}^{A_i} \bowtie \hat{\mathbb{D}}_{rx}^{A_l} \overset{k}{\underset{i=l+1}{\bowtie}} \mathbb{D}_{rx}^{A_i}.$$

The algorithm gives that

$$\hat{T}_k = T_k \cup M_k$$

where M_k can be expanded to

$$M_k = M_{k-1} \bowtie \mathbb{D}_{rx}^{A_k} = \ldots = M_l \overset{k}{\underset{i=l+1}{\bowtie}} \mathbb{D}_{rx}^{A_i}.$$

Since

$$M_l = T_{l-1} \bowtie \hat{\mathbb{D}}_{rx}^{A_l}$$

it follows that

$$M_k = T_{l-1} \bowtie \hat{\mathbb{D}}^{A_l}_{rx} \overset{k}{\underset{i=l+1}{\bowtie}} \mathbb{D}^{A_i}.$$

Inserting M_k in $\hat{T}_k$, and noticing that $T_{l-1} = \mathbb{D}^{\mathcal{A}} \bowtie_{i=1}^{l-1} \mathbb{D}^{A_i}_{rx}$ give that

$$\hat{T}_k = T_k \cup \mathbb{D}^{\mathcal{A}} \overset{l-1}{\underset{i=1}{\bowtie}} \mathbb{D}^{A_i}_{rx} \bowtie \hat{\mathbb{D}}^{A_l}_{rx} \overset{k}{\underset{i=l+1}{\bowtie}} \mathbb{D}^{A_i}_{rx}$$

which is what should be shown. The correctness of the update follows now by induction. □

5.4 SIMULATIONS

To test the algorithms, a hypothetical model of an embedded system has been constructed. It is inspired by the existing system described in Section 1.1.

5.4.1 Simulation Model

The model consists of components partitioned over four network-buses and including 20 ECUs. Components, connections, and the components supervised by the tests are chosen by random. Between 1 and 5 random faults have been inserted into the model.

A schematic picture of the model is shown in Figure 5.1. The number of components is between 200 and 2000 in the simulations performed here. The components are partitioned into three different levels, non-shared local components, components shared between the agents within only one bus, and components shared over the complete system. The number of components is given in percent of the total number of components. The number of outputs is about 5 % of the number of components, i.e. for 2000 components, there is about 100 outputs. The majority of the outputs are distributed within each net where each output is assigned to one agent and connected as input to other agents within the same net. The number of inputs is about 50 % higher than the number of outputs. The remaining outputs are assigned to any agent in the complete system and connected to any other agent in the complete system.

A single agent now consists of components, inputs, and outputs. Each agent is assigned tests that can detect faults in its components, which consist of the inputs and the local components. The number of tests is between 0.5 % to 5 % of the number of components, i.e. for 2000 components about 10 to 100 tests per agent. Due to the fact that most parts of the system is chosen by random, both the number and the complexity of the tests can wary substantially from agent to agent.

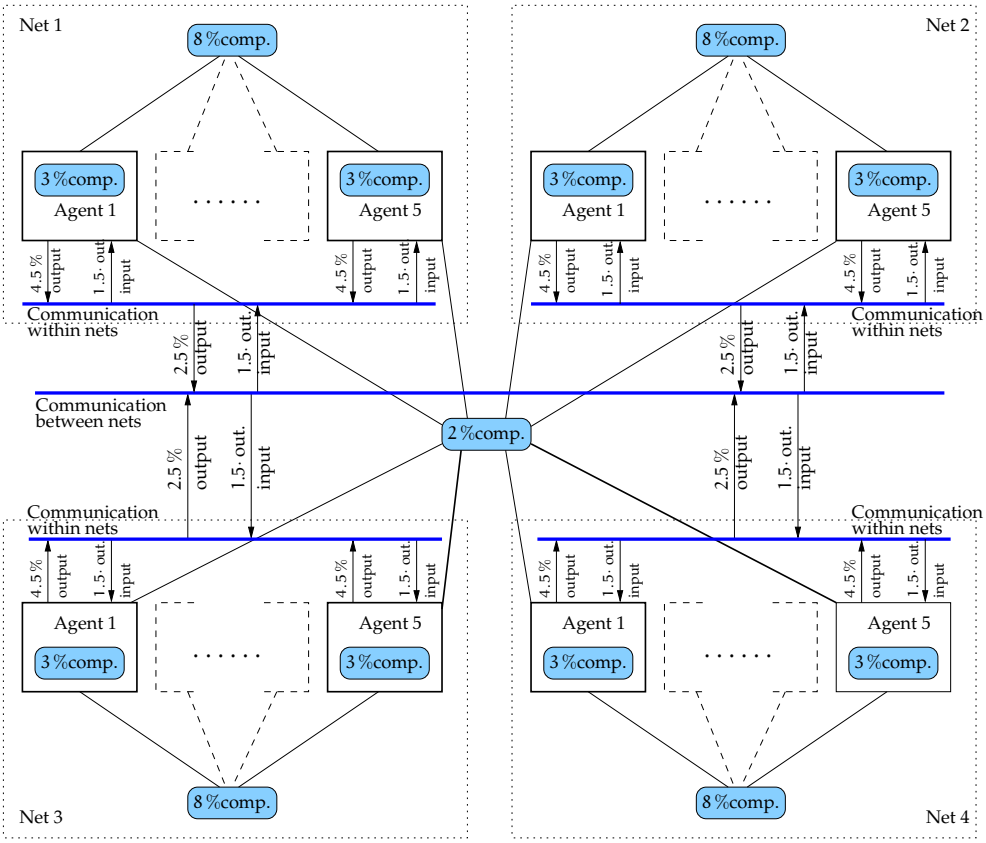


FIGURE 5.1: Simulation environment consisting of local components, components shared within a bus, and components shared over the complete system.

Now the model consists of components and diagnostic tests. Between 1 and 5 faults are inserted into the system and the tests have been given a 10 % detection probability. Conflicts are generated and the algorithms described in this chapter have been used to obtain the extended local diagnoses.

5.4.2 Transfer Times

There are several parts of the algorithms that take time to calculate. These are the time to choose which conflicts to transmit, the transmission of the conflicts, and the calculation of the extended diagnoses. The model is simulated in a computer and the time to transmit the conflicts is therefore negligible. To be able to correctly compare the methods, a transfer-time has been introduced in the model.

To simulate the transfer-time in the network, delays have been introduced whenever information is transferred over a bus. The delay is proportional to the cardinality of the conflicts and diagnoses, i.e. for a diagnosis D the delay is $k|D|$, where the constant $k = 0.005$.

5.4.3 Detection Degree

To compare the different algorithms, the detection degree for an agent is defined. If f is the set of faulty components, then for agent A the detection degree is

$$\begin{cases} \emptyset & \text{if } \nexists f_i \in \mathcal{C}^A \cup \mathrm{dep}(\mathrm{IN}^A) \\ \frac{|(f \cap (\mathcal{C}^A \cup \mathrm{dep}(\mathrm{IN}^A))) \cap \bigcup_{D \in \mathbb{D}^A} D|}{|f \cap (\mathcal{C}^A \cup \mathrm{dep}(\mathrm{IN}^A))|} & \text{otherwise.} \end{cases}$$

The detection degree returns $\emptyset$, if none of the faults is included in the local components or affects the local inputs by being included in the inputs dependencies. Otherwise, $|f \cap (\mathcal{C}^A \cup \mathrm{dep}(\mathrm{IN}^A))|$ is the number of faults that could affect the agent, and $\bigcup_{D \in \mathbb{D}^A} D$ are the set of components that have been detected by the minimal diagnoses. The number is normalized with the maximum number of detectable components and the detection degree is therefore a number between 1 and 0, where detection degree 1 means that all faults have been detected and 0 that none have been detected.

5.4.4 Hardware

The simulations have been performed on a 2.4 GHz personal computer running Linux Red hat [Red03] and Matlab [Mat05]. The simulation times are the processing time using Matlab's clocking capability.

Since the simulations have been performed on a single CPU, the parallel computing capabilities of the distributed system is not utilized. An approximation of the simulation time including transfer times for each agent is to divide the simulation time with the number of agents.

5.4.5 Limitations

Since most of the model is chosen by random, there is a possibility that a system is generated that is very complex. This could for example be that a very high number of inputs and outputs are included, or that the assumptions for some outputs are very large. Due to this complexity, it is sometimes not possible to simulate the system within reasonable times. These simulations are therefore removed from the simulations if the time to perform the simulation exceeds a time limit. The time limit is set to 2000 s. Notice that other limits, such as an limit proportional to the number of components, could be considered.

5.4.6 Simulations

Four different setups have been simulated. The first and the second are sharing of conflicts based on conflicts. The third and the fourth are sharing of conflicts based on the more likely diagnoses, where only the minimal cardinality diagnoses have been considered when choosing which conflicts to transmit.

In the first and the third have the complete conflicts been transmitted. In the second and the fourth have a virtual components been used. It has here been chosen to use only one virtual component to represent all the components in all the other agents. This to reduce the size of the diagnoses as much as possible. For reference, the calculation of the local diagnoses have been included in the plots.

The four setups have been simulated with 200 to 2000 components, with steps of 200 components. The mean values after 100 simulations are shown in Figure 5.2 to 5.5. In the figures, the new conflict to new conflict approaches are denoted by C2NC and C2NC, with Ω, respectively. The probable diagnoses to new conflict approaches are denoted by PD2NC, $\lim = 0$ and PD2NC, $\lim = 0$ with Ω, respectively.

5.4.7 Result

The simulation times are shown in Figure 5.2. Notice the exponential growth of the simulation times. It is clearly seen how the simulation time decreases when both the more probable diagnoses are used to chose which conflicts to share and when the virtual components are used.

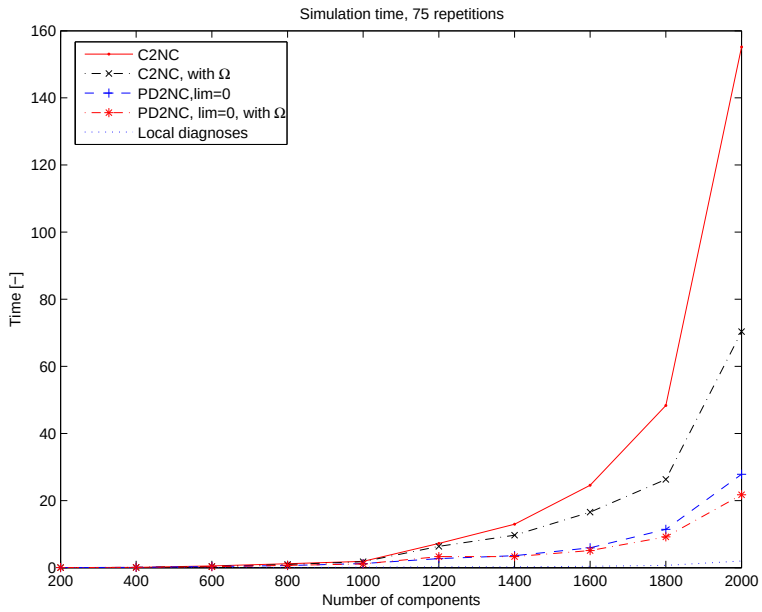


FIGURE 5.2: Simulation times.

The detection degree is shown in Figure 5.3. It is difficult to distinguish the two conflict to conflict approaches from each other. The same holds true for the two probable diagnoses to new conflicts approaches. At 1000 components and 2000 components it can be seen that the detection degree is slightly lower for the approach that use virtual components. There is a decrease in the detection degree when the probable diagnoses are used compared to when the conflicts are used to decide which conflicts to share. The plot also includes the detection degree for the union of all local diagnoses. This means that to each agent's local diagnoses, all other local diagnoses have been added. This gives an approximation of the upper limit of the detection degree. Together with the detection degree when only the original local diagnoses was included, this results in lower and upper limits. The plot shows that the detection degree is increased about halfway to the upper limit.

The mean number of diagnoses in each agent is shown in Figure 5.4. Notice how the approaches using virtual components have fewer diagnoses but, as was seen in Figure 5.3, this did not result in any major lowering in the detection degree. At 1200 and 1400 components, the conflict to conflict approach that use virtual component has a higher mean number of diagnoses.

Finally, the mean value of the cardinality for each diagnosis is shown

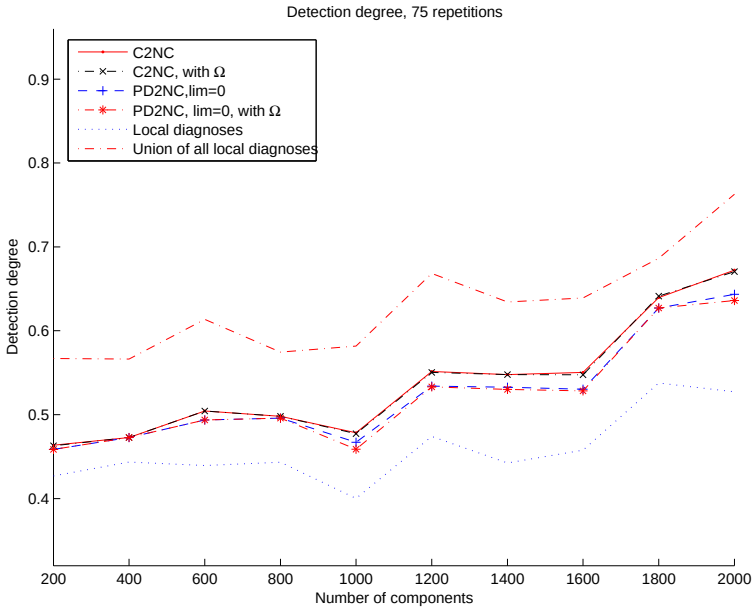


FIGURE 5.3: Detection degree.

in Figure 5.5. In mean, the cardinality of the algorithms including virtual components are larger. In these simulations, the mean size of the diagnoses is quite low, between 1.1 and 2.1 components.

To conclude: For these simulations, the simulation times decreases when the diagnoses are used to chose which conflicts to transmit, and when virtual components are used. The mean number of diagnoses in each agent also decreases, but not with any large degree. The detection degree decreases when the diagnoses is used to chose which conflicts to transmit, but with only a small degree when virtual components are used. The mean cardinality of each diagnosis increases when the virtual components are used, and when the diagnoses are used to decide which conflicts to transmit.

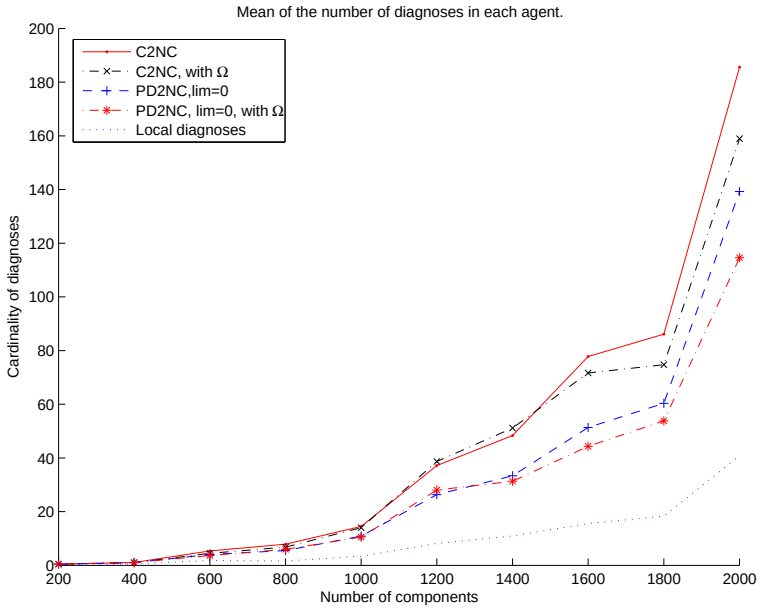


FIGURE 5.4: Mean number of diagnoses in each agent.

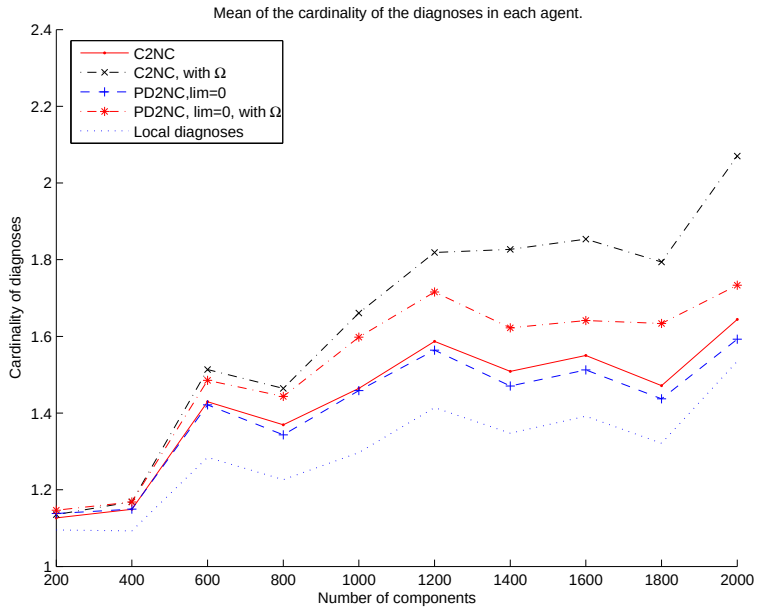


FIGURE 5.5: Mean of the cardinality of each diagnosis in each agent.

6

MINIMAL CARDINALITY GLOBAL DIAGNOSES

Due to the combinatorial growth of the number of global diagnoses when merging local diagnoses, it might be necessary to only calculate the more likely global diagnoses, i.e. the global diagnoses that more likely describe the current system behavior.

In this chapter, an algorithm is presented that calculates the *minimal cardinality module diagnoses* (MCMD). The algorithm can be run in a central diagnostic computer, or as presented here, it can distribute the computation intense tasks to the local ECUs. Often, there are limitations in processing power, memory, and network capacity, therefore, this possibility makes the algorithm more versatile. The MCMD property was discussed in Section 3.7.4.

The algorithm assumes that each agent has computed local diagnoses. The algorithm is limited to the case where the diagnoses are completely component represented, i.e. each diagnosis $D \subseteq \mathcal{C}$.

6.1 FINDING ALL MODULE MINIMAL CARDINALITY DIAGNOSES

After each agent has created local diagnoses, these could be merged to MCMDs. The MCMDs could be calculated in the following way: Divide the agents into modules such that the merge of the corresponding local diagnoses will become MCMDs; In each module, sort the agents into an

order such that the complexity of the upcoming merge is reduced, and then merge the local diagnoses according to this order.

A direct and simple approach to partition the agents into modules would be to first calculate

$$\bar{\mathbb{D}}^{\mathcal{A}} = \bigcup_{D \in \mathbb{D}^{\mathcal{A}}} D$$

for each agent, and then choose the minimal module $\bar{\mathcal{A}}$ such that

$$\bigcup_{\mathcal{A} \in \bar{\mathcal{A}}} \bar{\mathbb{D}}^{\mathcal{A}}$$

is disjoint from all other such sets. A simple ordering is to choose the agent with largest minimal cardinality diagnosis to be first and then the rest follows in decreasing cardinality.

A direct and simple merge can be done by firstly finding a low lower limit on the size of the MCMDs and then merge those of the local diagnoses that are smaller or equal to this limit. Notice that those that are larger cannot be a part of a MCMD. If no MCMDs was found, then the limit is increased and the merge is started again from the first agent.

However, the computation time of this approach can be greatly improved by further reducing the size of the modules, sorting them into a better order, and finally merging the diagnoses such that the total number of merges is minimized. These improvements have been implemented in the algorithm described in this section.

6.2 MAIN ALGORITHM

The improved algorithm is shown on page 89 and consists of the same three main parts as the approach described above. Firstly, algorithm $\mathrm{FindSubGraph}$ is used to find a graph $\mathbb{G}$ whose *sub graphs represent the modules*. Algorithm FindR is then used to sort the agents in a sub graph into a *merge order* R. The agents in the module are ordered in such a way that the complexity of the remaining algorithm is reduced. Finally, the local diagnoses are with algorithm $\mathrm{UpdateAgent}$ iteratively merged into sets of MCMDs.

The improvements over the first approach are that the modules are partitioned into smaller sets, the merge orders are chosen in a better way, and that the merge algorithm is much more efficient because it both keeps track and uses what have been done in previous runs.

Notice that the efficiency of the merge is highly dependent on the diagnoses, meaning that sometimes the new ordering is not better than the first, and sometimes it might actually be worse. Over a period

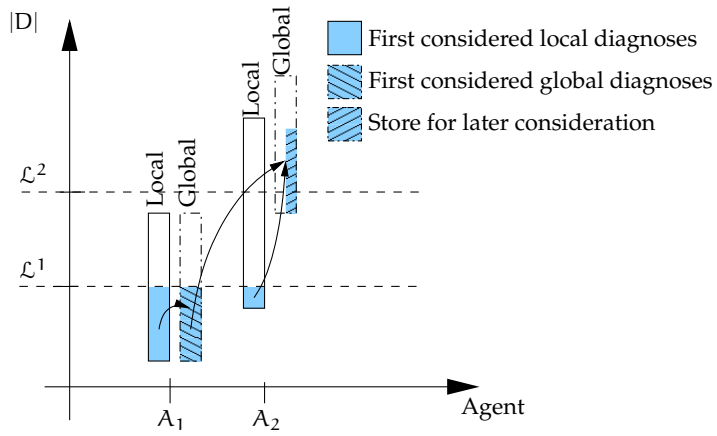


FIGURE 6.1: Schematic merge of the local diagnoses in two agents. The first iteration.

of times, it is however likely that this approach is more efficient. The module division and the merge are always more efficient, not counting the computation time for the more complex algorithm.

The algorithm can be *distributed* in such a way that Algorithm 4, FindSubGraph, and FindR are evaluated in some coordinating agent, while the computation and memory intensive UpdateAgent is *evaluated in the corresponding agent*.

6.2.1 Outline of The Algorithm

The following example will be used to introduce the outline of the algorithm.

EXAMPLE 6.1: The ideas behind the algorithm are shown schematically in Figure 6.1 and 6.1 for a two agent system. The rectangles represent local and global diagnoses with increasing cardinality.

The algorithm starts with a lower limit $\mathcal{L}^1$. In 6.1, the local diagnoses with cardinality lower than the limit in A_1 and A_2 are considered when computing global diagnoses. The local diagnoses in A_1 are transferred to global diagnoses in A_1 and then merged with the local diagnoses in A_2. Since the cardinality of the global diagnoses in A_2 are greater than the limit, the limit is increased and the computation is started again from agent A_1.

The limit has been increased in Figure 6.2. The local diagnoses in A_1 that were not considered in the previous computation are now transferred to global diagnoses in A_1. The new global diagnoses in

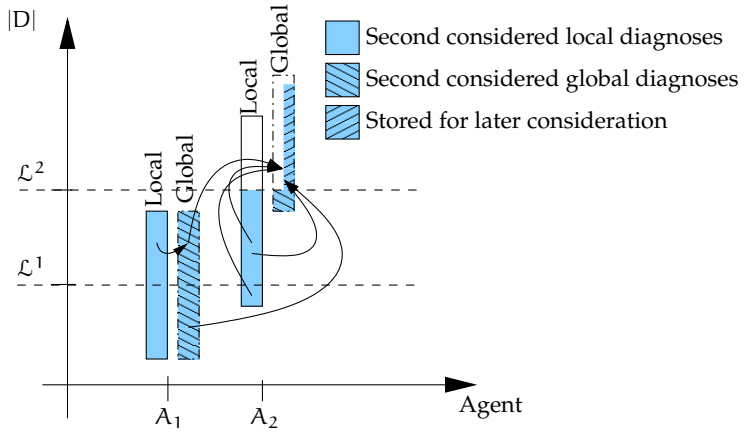


FIGURE 6.2: Schematic merge of the local diagnoses in two agents. The second iteration.

A_1 are merged with all considered local diagnoses in A_2. There might also exist global diagnoses that comes from the old global diagnoses in A_1 merged with the newly considered local diagnoses in A_2, therefore these are also merged.

Since some global diagnoses have been found with cardinality lower than the limit, the algorithm is terminated. The global diagnoses are the minimal cardinality diagnoses. ◇

Now consider the main Algorithm 4 where an ordered set is represented by $(\cdot)$ and an unordered set by $\{\cdot\}$. Firstly, the graph is found and a merge order R is calculated from each sub graph. The algorithm starts with a low lower limit $\mathcal{L}$ on the cardinality of the MCMDs. The first approximation of $\mathcal{L}$ is that the MCMDs must be at least as large as the largest minimal cardinality local diagnosis, considering all agents in R. After the limit has been found, an evaluation of $\mathrm{UpdateAgent}$ is performed in each agent or until an agent returns a new value on $\mathcal{L}$.

Algorithm $\mathrm{UpdateAgent}$ calculates those diagnoses with cardinality less than or equal to $\mathcal{L}$ that would be formed if the local diagnoses in agents A_1 to A_k, which also have a cardinality less than or equal to $\mathcal{L}$, was merged. The set is denoted N in the algorithm. If no diagnoses could be found with cardinality less than or equal to $\mathcal{L}$, a new larger $\mathcal{L} = \mathcal{L}^{new}$ is calculated and a new search begins from A_1. This new $\mathcal{L}$ is chosen so that new merged diagnoses will be found at the next iteration. When the last agent calculates a non-empty N, this set of

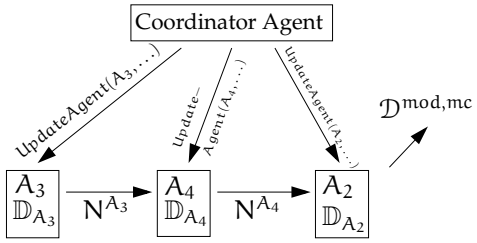


FIGURE 6.3: The information flow in Example 6.2.

Algorithm 4 Minimal cardinality module diagnoses

Require: All local diagnoses $\mathbb{D}^{\mathcal{A}}$ for all agents $\mathcal{A}$.
Ensure: All MCMDs $\mathcal{D}_m^{mod,mc}$.

1: $\mathbb{G} := \mathrm{FindSubGraph}(\mathcal{A})$ [*Graph of sub graphs.*]
2: **for all** $G \in \mathbb{G}$ **do**
3: $R := \mathrm{FindR}(G)$ [*Merge order* $R = (A_1, \ldots, A_n)$.]
4: $\mathcal{L} :=$ a lower bound on the cardinality for the MCMDs in R.
5: $k := 0$, $A_0 := \emptyset$
6: **repeat**
7: $k := k + 1$
8: $\mathcal{L}^{new} := \mathrm{UpdateAgent}(A_k, A_{k-1}, \mathcal{L})$
[N *stored in* A_k *includes the new diagnoses.*]
9: **if** $\mathcal{L}^{new} > \mathcal{L}$ **then**
10: $k := 0$, $\mathcal{L} := \mathcal{L}^{new}$ [*Increase and restart.*]
11: **end if**
12: **until** $k = n$ [A_n *is the last in* R.]
[$\mathcal{D}_m^{mod,mc} =$ N *stored in* A_n, *for the m:th* R.]
13: **end for**

diagnoses is the MCMDs for this module.

EXAMPLE 6.2: Consider Figure 6.3 where the MCMDs should be found for the merge order (A_3, A_4, A_2). Algorithm 4 sends the $\mathrm{UpdateAgent}$ command with input $(A_3, \emptyset, \mathcal{L})$ to agent A_3, which extracts the diagnoses with cardinality lower than or equal to $\mathcal{L}$, denoted N^{A_3}. Thereafter, the $\mathrm{UpdateAgent}$ command is sent to A_4, which collects N^{A_3} over the network and merge it with its own local diagnoses with cardinality less than or equal to $\mathcal{L}$; those of the merged diagnoses with cardinality less than or equal to $\mathcal{L}$ is stored in N^{A_4}. After some iterations, the end result is the MCMDs $\mathcal{D}^{mod,mc}$.

◇

6.3 FIND SUB GRAPH $\mathbb{G}$ – ALGORITHM 5

In the direct approach the agents was partitioned with respect to all components in all diagnoses. This is however a pessimistic approach since some of the diagnoses might have cardinality higher than the cardinality of the MCMDs, which means that it is unnecessary to consider them when deciding the modules.

The main idea in Algorithm 5, see page 92, is to find an upper limit $\mathcal{U}$ of the cardinality of the MCMDs, and use this to reduce the size of the modules.

A first value of the upper limit can be chosen as

$$\mathcal{U} := \sum_{A \in \bar{\mathcal{A}}} \min_{D \in \mathbb{D}^A} |D|$$

which is the sum of the minimal cardinality local diagnoses. A lower upper limit is found by making a pre-merge of all minimal cardinality diagnoses. To reduce the complexity it is possible to only consider the minimal cardinality diagnoses after each merge, which would give

$$\mathcal{U} := \min_{D \in \bar{\mathbb{D}}} |D|$$

where

$$\bar{\mathbb{D}} := (\ldots((\mathbb{D}^{mc}_{A_1} \bowtie \mathbb{D}^{mc}_{A_2})^{mc} \bowtie \mathbb{D}^{mc}_{A_3})^{mc} \ldots).$$

This later limit would however require more computations than the first. The first limit, i.e. the sum of the minimal cardinalities, will be used in the examples and in the simulations described last in the chapter.

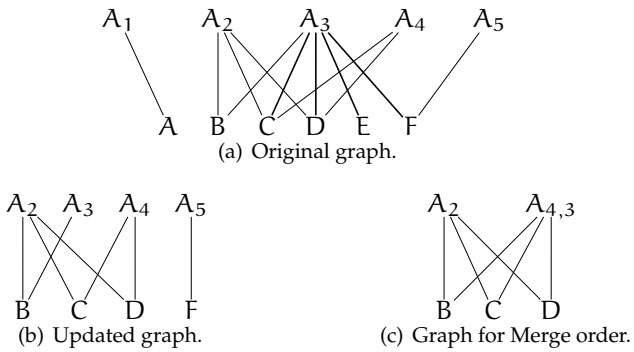
For each sub graph found, Algorithm 5 is called recursively so that each sub graph is, if it is possible, partitioned into even smaller sub graphs.

EXAMPLE 6.3: Consider a system with five agents including the local diagnoses

$$\mathbb{D}^1 = \{\{A\}\} \qquad \mathbb{D}^2 = \{\{B, C\}, \{D\}\}$$
$$\mathbb{D}^3 = \{\{B\}, \{C, D, E, F\}\} \qquad \mathbb{D}^4 = \{\{C\}, \{D\}\}$$
$$\mathbb{D}^5 = \{\{F\}\}$$

When using Algorithm 5, $\bar{\mathbb{D}} = \{\{A, D, B, F\}\}$ and therefore the upper limit is $\mathcal{U} = 4$. The graph shown in Figure 6.4(a) is created. It is partitioned into two sub graphs which corresponding sets of agents are used in the next iteration. The first sub graph only includes agent A_1.

For the second sub graph, the limit is $\mathcal{U} = 3$ and the diagnosis $\{C, D, E, F\}$ can therefore be emitted, since it can not be included in an



(a) Original graph.

(b) Updated graph.

(c) Graph for Merge order.

FIGURE 6.4: The graph representing a system with five agents and six components.

MCMD. The result is the graph shown in Figure 6.4(b) that is further partitioned into two sub graphs. The final result is that the agents are partitioned into three sub graphs including $\{A_1\}$, $\{A_2, A_3, A_4\}$, and $\{A_5\}$. ◇

6.4 FINDING THE MERGE ORDER $\mathbb{R}$ – ALGORITHM 6

The calculation of a merge order R is an interesting problem that can be solved with Algorithm 6, see page 93. By changing the merge order, the complexity of Algorithm 7 might change dramatically. The input is one of the sub graphs that were found with Algorithm 5. The output is an ordered set $R \subseteq \mathcal{A}$, where each R represents a module.

When calculating R, the main idea is that the lower bound $\mathcal{L}$ in Algorithm 4 should be raised as soon as possibly towards its final value. This to reduce the complexity when the local diagnoses finally are merged. To raise $\mathcal{L}$ as fast as possible, the sets of local diagnoses should have few components in common and the minimal cardinality local diagnoses should be large. To compromise between the number of common components c and the cardinality r of the diagnoses is function $\alpha(c, r)$ used. Based on simulation results, the function

$$\alpha(c, r) = (c + 5)/r^2$$

is used in the simulations in the end of the chapter.

The number of common components is decided by looking at the number of connections between the agents and the components in the

Algorithm 5 FindSubGraph

Input: Set of agents $V^{\mathcal{A}} \subseteq \mathcal{A}$. Require the local diagnoses $\mathbb{D}^{\mathcal{A}}$ stored in the local agents.

Output: The graph $\mathbb{G}$ consisting of sub graphs G, where a sub graph $G = (\bar{V}^{\mathcal{A}}, \bar{V}^{\mathcal{C}}, \bar{E})$.

1: $\mathcal{U}$:= a upper bound on the cardinality for the MCMDs.
2: $\bar{\mathbb{D}}^{\mathcal{A}} := \{D \mid D \in \mathbb{D}^{\mathcal{A}}, |D| \leqslant \mathcal{U}\}$ for each agent
3: $V^{\mathcal{A}} := \{V^{\mathcal{A}} \mid \bar{\mathbb{D}}^{\mathcal{A}} \neq \emptyset\}$
4: $V^{\mathcal{C}} := \bigcup_{A \in V^{\mathcal{A}}} \bigcup_{D \in \bar{\mathbb{D}}^{A}} D$
5: $E := \{(A, c) \mid A \in V^{\mathcal{A}}, c \in \bigcup_{D \in \bar{\mathbb{D}}^{A}} D\}$
6: $\mathbb{G} := (V^{\mathcal{A}}, V^{\mathcal{C}}, E)$ [*Graph.*]
7: Find sub graphs $G \in \mathbb{G}$.
8: **if** $|\mathbb{G}| > 1$ **then** [*More than 1 sub graph.*]
9: $\mathbb{G} := \{\text{FindSubGraph}(\bar{V}^{\mathcal{A}}) \mid \bar{V}^{\mathcal{A}} \in G, G \in \mathbb{G}\}$ [*Recursive.*]
10: **end if**

graph. Function Γ is used for this; the function gives the number of components that are shared between two agents

$$\Gamma(A_1, A_2) = |\{c \mid (A_1, c) \in E, (A_2, c) \in E\}|.$$

After two sets of local diagnoses have been merged, the merged set might include components from both sets of local diagnoses. Therefore, the agents and the corresponding components should be joined in the graph. A join of A_2 to A_1 in graph $G = (V^{\mathcal{A}}, V^{\mathcal{C}}, E)$

$$\text{join}(G, A_1, A_2) = (V^{\mathcal{A}} \backslash A_2, V^{C}, \bar{E})$$

where

$$\bar{E} = E \backslash \{(A_2, c) \mid \forall c\} \cup \{(A_1, c) \mid (A_2, c) \in E\}.$$

EXAMPLE 6.4: Consider the first sub graph in Example 6.3 and Figure 6.4(b). Using the algorithm with the sub graph as input, finds that A_3 and A_4 have the least number of components in common, i.e. zero components which give $\alpha(A_3, A_4) = 1$; since this is the lowest weight, $R = (A_3, A_4)$. After this A_3 is merged to A_4 which gives the graph in Figure 6.4(c). Agent A_2 has three components in common with $A_{4,3}$ and is added to R, which give $R = (A_3, A_4, A_2)$. ◇

6.5 UPDATE AGENT – ALGORITHM 7

Function UpdateAgent is computed with Algorithm 7, see page 94. When the main algorithm should evaluate UpdateAgent with input

Algorithm 6 FindR

Input: sub graph $G(V^{\mathcal{A}}, V^{c}, E)$.
Output: A merge order R which is an ordering of $V^{\mathcal{A}}$.

if $|V^{\mathcal{A}}| = 1$ **then**
  $R := (V^{\mathcal{A}})$ [*The vertex* $V^{\mathcal{A}} \in \mathcal{A}$.]
else
  $(A_i, A_j) \in \arg\min_{A_i, A_j} \alpha(\Gamma(A_i, A_j), \min_{D \in \mathbb{D}^{A_i}} |D|)$
  $R := (A_i, A_j)$ [*First ordered set.*]
  $G := \text{join}(G, A_j, A_i)$ [A_i *merged* A_j.]
  while $|R| < |V^{\mathcal{A}}|$ where $R = \{A_1, \ldots, A_{end}\}$ **do**
    $A \in \arg\min_A \alpha(\Gamma(A, A_{end}), \min_{D \in \mathbb{D}^A} |D|)$
    $G := \text{join}(G, A, A_{end})$ [A_{end} *joined* A.]
    $R := R \cup A$ [*Ordered set. New* $A_n = A$.]
  end while [*All* $A \in V^{\mathcal{A}}$ *is included in* R.]
end if

$(A_k, A_{k-1}, \mathcal{L})$, it sends this command to agent A_k. The agent that receives this command should find the diagnoses with cardinality less than or equal to $\mathcal{L}$ in the set $\bowtie_{A=A_1}^{A_k} \mathbb{D}^A$. Since agent A_{k-1} has already calculated the diagnoses with cardinality less than or equal to $\mathcal{L}$ from $\bowtie_{A=A_1}^{A_{k-1}} \mathbb{D}^A$, the desired diagnoses can be calculated from the result in A_{k-1} merged with $\mathbb{D}^{A_k}$.

In the direct approach, this is repeated each time that $\mathcal{L}$ is raised. Algorithm 7 is more efficient, since it only calculates those diagnoses that have not been considered in previous runs, i.e. those with cardinality between the old and the new $\mathcal{L}$.

The main parts of the algorithm are the merge of the *old and new global diagnoses* for $\{A_1, \ldots, A_{k-1}\}$ with the *new local diagnoses* (TE), and the merge of the *new global diagnoses* for $\{A_1, \ldots, A_{k-1}\}$ with the *old local diagnoses* (TF). The merged diagnoses with cardinality greater than $\mathcal{L}$ are stored in M for later consideration, i.e. if $\mathcal{L}$ is later raised then parts of M might be included in the new N. Variable l is the number of times that the current agent has been called with command $\text{UpdateAgent}(\cdot)$.

The new global diagnoses N is saved for use by agent A_{k+1}. If A_k is the last agent and $N \neq \emptyset$ then N is the set of MCMDs, i.e. the wanted result.

EXAMPLE 6.5: Consider the second set of agents found in Example 6.4, with the sort order $R = (A_3, A_4, A_2)$. The local diagnoses for these

Algorithm 7 UpdateAgent

Input: A_k, A_{k-1} and $\mathcal{L}$. Require $\mathbb{D}^{A_k}$ stored in agent A_k and N stored in agent A_{k-1}.

Output: New sub-set of $\mathcal{D}^{mod}$ for agents $\{A_1, \ldots, A_k\}$ with cardinality $\leqslant \mathcal{L}$ stored as N^{A_k} in agent A_k. $\mathcal{L}$ as output.

1: $E := \{D \,|\, |D| \leqslant \mathcal{L}, D \in \mathbb{D}\}$
2: $\mathbb{D} := \mathbb{D} \backslash E$
3: **if** $i = 1$ **then** [*The first A in R.*]
4: $N := E$ [*New diagnoses from A_k.*]
5: **else if** $i > 1$ **then**
6: $T_l := N$ in A_{k-1} [*New diagnoses from A_{k-1}.*]
7: $ET := \bigcup_{j=1}^{l} E \bowtie T_j$ [*Old and new merged with new.*]
8: $FT := F \bowtie T_l$ [*New merged with old.*]
9: $\mathbb{D}_{new} := ET \cup FT \cup M$
10: $N := \{D \,|\, |D| \leqslant \mathcal{L}, D \in \mathbb{D}_{new}\}$ [*New from A_k.*]
11: $M := \mathbb{D}_{new} \backslash N$ [*Store for later consideration.*]
12: **if** $N = \emptyset$ and N has not been non-empty **then**
13: $\mathcal{L} := \min(\min_{m \in M} |m|, \min_{D \in \mathbb{D}} |D|)$ [*New lower limit.*]
14: **end if**
15: $F := F \cup E$
16: **end if**

agents are

$$\mathbb{D}^2 = \{\{B, C\}, \{D\}\} \qquad \mathbb{D}^3 = \{\{B\}, \{C, D, E, F\}\}$$
$$\mathbb{D}^4 = \{\{C\}, \{D\}\}$$

With $\mathcal{L} = 1$, the first agent A_3 finds the new diagnoses with cardinality less than or equal to $\mathcal{L}$ which give $N^{A_3} = \{\{B\}\}$. Agent A_4 collects N^{A_3} over the network and calculates $W^{A_4} := \{\{B, C\}, \{B, D\}\}$, but since $\mathcal{L} = 1$, the new set of diagnoses $N^{A_4} := \emptyset$. Since $N^{A_4} = \emptyset$ and N^{A_4} has not been non-empty, a new $\mathcal{L}$ is $\mathcal{L}^{new} = 2$. Since a new $\mathcal{L}$ was returned, the algorithm starts over from the first agent. A_3 has already stated diagnosis $\{B\}$ and since there are no other diagnoses with low cardinality, the new diagnoses are $N^{A_3} := \{\{\}\}$. Agent A_4 collects the new diagnoses and finds the new diagnoses $N^{A_4} := \{\{B, C\}, \{B, D\}\}$. A_2 takes over and calculates the new diagnoses $N^{A_2} := \{\{B, C\}, \{B, D\}\}$. Since $N^{A_n} = N^{A_2} \neq \emptyset$, the iterations ends and the set of MCMDs is $\mathcal{D}^{mod,mc} = N^{A_2}$, which is stored in agent A_2. ◇

6.6 CORRECTNESS OF THE ALGORITHMS

The result of Algorithm 4 is given by Proposition 6.1.

PROPOSITION 6.1 (Correctness of the algorithm): *The result of Algorithm 4 is all sets of* MCMDs

$$\mathcal{D}_k^{mod,mc}$$

considering the local diagnoses $\mathbb{D}^A$ *for all agents* $A \in \mathcal{A}$.

The proposition needs some lemmas to be proven. The first lemma states that the different sub graphs from Algorithm 4 will truly represent modules.

LEMMA 6.2 (Modules): *Let* $\mathbb{G} = \text{FindSubGraph}(\mathcal{A})$, *where each* $G^k \in \mathbb{G}$ *is a sub graph* $G^k = (V_k^{\mathcal{A}}, V_k^{\mathcal{C}}, E_k)$, *and let*

$$\bar{\mathbb{D}}^A = \{D \,\big|\, |D| \leqslant \mathcal{U}, D \in \mathbb{D}^A\}$$

for the last $\mathcal{U}$ *that partitioned* G_k *into a sub graph. Then* $V_k^{\mathcal{A}}$ *is a module considering the diagnoses* $\bar{\mathbb{D}}^A$.

Note that the modules are defined considering the sets $\bar{\mathbb{D}}^A$. The modules cold thereby be smaller than the modules that could be found when considering the complete set of diagnoses $\mathbb{D}^A$.

Proof. From Algorithm 5 and the definition of sub graphs follows directly that $D_1 \cap D_2 = \emptyset$ and $V_1^A \cap V_2^A = \emptyset$, for two sets of V_k^A, where $D_k \in \bowtie_{A \in V_k^A} \bar{\mathbb{D}}^A$. □

LEMMA 6.3 (MCMDs): *Same set-up as in Lemma 6.2. Let* $V_k^{\mathcal{A}}$ *be a module, then the minimal cardinality module diagnoses*

$$\bar{\mathcal{D}}_k^{mod,mc} = \mathcal{D}_k^{mod,mc}$$

where

$$\bar{\mathcal{D}}_k^{mod,mc} = \{D \,\big|\, |D| = \min_{D \in \mathcal{D}} |D|, D \in \mathcal{D}, \mathcal{D} = \bowtie_{A \in V_k^A} \bar{\mathbb{D}}^A\}$$

$$\mathcal{D}_k^{mod,mc} = \{D \,\big|\, |D| = \min_{D \in \mathcal{D}} |D|, D \in \mathcal{D}, \mathcal{D} = \bowtie_{A \in V_k^A} \mathbb{D}^A\}.$$

Proof. Each diagnosis $D \in \bowtie_{A \in V_k^A} \mathbb{D}^A$ is a set $D = \bigcup_{A \in V_k^A} D^A$ where each $D^A \in \mathbb{D}^A$. The cardinality of each such diagnosis is

$$|D| \geqslant \max_{A \in V_k^A, D^A \in \mathbb{D}^A} |D^A|.$$

The upper limit is defined so that for $D \in \mathcal{D}_k^{mod,mc}$, the cardinality is

$$|D| \leqslant \mathcal{U}.$$

If the cardinality of a diagnosis is $|D| > \mathcal{U}$ then $D \notin \mathcal{D}_k^{mod,mc}$. Together, this means that the set $\bar{\mathcal{D}}_k^{mod,mc} \supseteq \mathcal{D}_k^{mod,mc}$. Since $\bar{\mathcal{D}}_k^{mod,mc} \subseteq \mathcal{D}_k^{mod,mc}$, it follows that $\bar{\mathcal{D}}_k^{mod,mc} = \mathcal{D}_k^{mod,mc}$. □

The next lemma states that diagnoses of a certain cardinality are found and if no diagnoses were found then $\mathcal{L}_{new}$ will have a specific value.

LEMMA 6.4 (Cardinality of diagnoses): *Given* $R = \{A_1, \ldots, A_k\}$ *and* $\mathcal{L}_n$. *Let the set of different* $\mathcal{L}$ *be* $\bar{\mathcal{L}} = \{\mathcal{L}_{init}, \mathcal{L}_1, \ldots, \mathcal{L}_{n-1}, \mathcal{L}_n\}$, *where the present* $\mathcal{L}$ *is* $\mathcal{L}_n$. *After evaluating* $\mathcal{L}_{new} = \mathrm{UpdateAgent}(A_k, A_{k-1}, \mathcal{L}_n)$, *then* N *in agent* A_k *is*

$$N^{A_k} = \{D \,\big|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathcal{D}\}$$

where $\mathcal{D} = \bowtie_{A \in R} \mathbb{D}^A$.

Proof. The lemma will be proven by induction. If $k = 1$, i.e. $R = A_1$, then by direct use of the algorithm follows that A_1 will include

$$N^{A_1} := E = \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathcal{D}\}$$

where $\mathcal{D} = \mathbb{D}^{A_1}$.

Let

$$\bar{\mathbb{D}}_i^A = \{D \,\big|\, |D| \leqslant \mathcal{L}_i, D \in \mathbb{D}^A\}$$

i.e. the set of local diagnoses with cardinality less than or equal $\mathcal{L}_i$.

For $k > 1$ and $n > 1$, let $\bar{R} = \{A_1, \ldots, A_{k-1}\}$ and consider agent A_k. Suppose that $N^{A_{k-1}}$ is correct, i.e.

$$N^{A_{k-1}} = \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \underset{A \in \bar{R}}{\bowtie} \mathbb{D}^A\},$$

and that $\mathrm{UpdateExtendedDiagnoses}(A_k, A_{k-1}, \mathcal{L}_n)$ is evaluated for the first time for agent A_k, i.e. $l = 1$. Since it is the first evaluation, the algorithm gives the new set

$$E = \bar{\mathbb{D}}_n^{A_k}.$$

Received from A_{k-1} is the set $T_l = N^{A_{k-1}}$. From the algorithm follows that

$$\mathbb{D}^{new} = E \bowtie T_1$$

since $F = \emptyset$ and $M = \emptyset$. Therefore, the set N^{A_k} is

$$N^{A_k} = \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \underset{A \in R}{\bowtie} \mathbb{D}^A\}.$$

By induction follows now that the first N^{A_k}, i.e. $l = 1$, is correct for all k and n.

For $k > 1$, $n > 1$, and $l > 1$. Suppose that $N^{A_{k-1}}$ for agent A_{k-1} is the correct diagnoses consistent with all diagnoses from the

agents in $\bar{R}$. Further suppose that the previous evaluations of $\mathrm{Update\text{-}ExtendedDiagnoses}(A_k, A_{k-1}, \mathcal{L}_i)$ for A_k are correct. Consider now $\mathrm{UpdateExtendedDiagnoses}(A_k, A_{k-1}, \mathcal{L}_n)$ for evaluation l.

The above means that $T_1, \ldots, T_{l-1}$, M, F, and $\mathbb{D}$ in A_k are correct. The following information is stored in agent A_k from previous runs

$$T_i = \{D \,|\, \mathcal{L}_{n+i-l-1} < |D| \leqslant \mathcal{L}_{n+i-l}, D \in \underset{A \in \bar{R}}{\bowtie} \bar{\mathbb{D}}^A_{n+i-l}\}$$

for $i \in \{1, \ldots, l-1\}$,

$$M = \{D \,\big|\, |D| > \mathcal{L}_{n-1}, D \in \underset{A \in R}{\bowtie} \bar{\mathbb{D}}^A_{n-1}\}$$
$$F = \mathbb{D}^{A_k}_{n-1}$$
$$\mathbb{D} = \mathbb{D}^{A_k} \backslash F.$$

The algorithm gives the new

$$E := \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathbb{D}\}$$

Since only those diagnoses with cardinality greater than $\mathcal{L}_{n-1}$ have been merged in the previous agents, the diagnoses received from A_{k-1} is

$$T_l := \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \underset{A \in \bar{R}}{\bowtie} \bar{\mathbb{D}}^A_n\}.$$

The sets included in $\mathbb{D}^{new}$ are M, FT, and ET, where

$$M := \{D \,\big|\, |D| > \mathcal{L}_{n-1}, D \in \underset{A \in R}{\bowtie} \bar{\mathbb{D}}^A_{n-1}\}$$
$$FT := \mathbb{D}^{A_k}_{n-1} \bowtie \{D \,\big|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \underset{A \in \bar{R}}{\bowtie} \bar{\mathbb{D}}^A_n\}$$
$$ET := \{D \,\big|\, |D| \leqslant \mathcal{L}_n, D \in \underset{A \in \bar{R}}{\bowtie} \bar{\mathbb{D}}^A_n\} \bowtie \{D \,\big|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathbb{D}^{A_k} \backslash \mathbb{D}^k_{n-1}\}.$$

Set ET is calculated from E and the union of the T_i:s, where

$$\bigcup_{j=1}^{l} T_j := \{D \,\big|\, |D| \leqslant \mathcal{L}_n, D \in \underset{A \in \bar{R}}{\bowtie} \bar{\mathbb{D}}^A_n\}.$$

The diagnoses $\mathcal{D} = \bowtie_{A \in R} \mathbb{D}^A$ can be partitioned into different sets with respect to the cardinality of the diagnoses from agent A_k and $\bar{R}$ respectively. Such a partition of the set if

$$\mathcal{D} = M \cup ET \cup FT \cup H$$

where

$$H = \{D \,\big|\, |D| > \mathcal{L}_n, D \in \underset{A \in \bar{R}}{\bowtie} \bar{\mathbb{D}}^A_n\} \bowtie \{D \,\big|\, |D| > \mathcal{L}_n, D \in \mathbb{D}^{A_k}\}.$$

TABLE 6.1: Sets of diagnoses. To the left and on the top are two different sets of diagnoses partitioned over different cardinalities.

		$D \in \bowtie_{A \in \bar{R}} \mathbb{D}^A$		
$D \in \mathbb{D}^{A_k}$	$\bowtie$	$\lvert D\rvert \leqslant \mathcal{L}_{n-1}$	$\mathcal{L}_{n-1} < \lvert D\rvert \leqslant \mathcal{L}_n$	$\lvert D\rvert > \mathcal{L}_n$
	$\lvert D\rvert \leqslant \mathcal{L}_{n-1}$	M	FT	H
	$\mathcal{L}_{n-1} < \lvert D\rvert \leqslant \mathcal{L}_n$	ET	ET	H
	$\lvert D\rvert > \mathcal{L}_n$	H	H	H

Table 6.1 shows which parts of the diagnoses that are found in variables M, ET, FT, and H respectively.

From the algorithm is given that the set N^{A_k} is

$$\begin{aligned} N^{A_k} &= \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathbb{D}^{new}\} \\ &= \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in M \cup ET \cup FT \cup H\} \end{aligned}$$

since $|D| > \mathcal{L}_n$ for $D \in H$. Therefore

$$N^{A_k} = \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathcal{D}\}.$$

By induction follows now that N^{A_k} is as stated in the proposition for all n, all l, and all k. □

The next lemma states the result after the final call from Algorithm 4 to Algorithm 7 for a set of agents R.

LEMMA 6.5 (Result of UpdateAgent): *If* $N^{A_n} \neq \emptyset$ *in agent* A_n *after evaluation of* $\mathrm{UpdateAgent}(A_n, A_{n-1}, \mathcal{L})$*, where* $R = \{A_1, \ldots, A_n\}$*, then*

$$N^{A_n} = \mathcal{D}^{mc}$$

where $\mathcal{D} = \bowtie_{A \in R} \mathbb{D}^A$.

Proof. After an update with $N^{A_n} \neq \emptyset$, the lower limit could have been decided by A_n or a preceding agent A_p in R. If A_n has decided the limit, then $|D| \geqslant \mathcal{L}_n$ for $D \in \bowtie_{A \in R} \mathbb{D}^A$. If an earlier agent has decided, then $|D| \geqslant \mathcal{L}$ for $D \in \bowtie_{A \in \{A_1, \ldots, A_p\}} \mathbb{D}^A$. This later limit also results also means that $|D| \geqslant \mathcal{L}_n$ for $D \in \bowtie_{A \in R} \mathbb{D}^A$.

From Lemma 6.4 follows that

$$N^{A_k} = \{D \,|\, \mathcal{L}_{n-1} < |D| \leqslant \mathcal{L}_n, D \in \mathcal{D}\}$$

where $\mathcal{D} = \bowtie_{A \in R} \mathbb{D}^A$. Since $|D| \geqslant \mathcal{L}_n$, and $\mathcal{L} = \min_{D \in \bowtie_{A \in R} \mathbb{D}^A} |D|$, it follows that

$$N^{A_k} = \{D \,\big|\, |D| = \min_{D \in \mathcal{D}} |D|, D \in \mathcal{D}, \mathcal{D} = \underset{A \in R}{\bowtie} \mathbb{D}^A\}$$

and by definition of minimal cardinality follows that

$$N^{A_n} = \mathcal{D}^{mc}.$$

□

Finally, the proof of Theorem 6.1.

Proof. Lemma 6.2 showed that V_k^A from Algorithm 4 is a module, and Lemma 6.3 showed that $\mathcal{D}_k^{mod,mc}$ is a set of minimal cardinality module diagnoses considering the set of diagnoses $\bar{\mathbb{D}}^A$. The sort algorithm FindR does not affect the agents included in the set given from $\mathrm{FindSubGraph}$. Finally, Lemma 6.5 showed that the result from Algorithm 7 is a set of minimal cardinality diagnoses. Thereby is the set of all $\mathcal{D}_k^{mod,mc}$ the MCMDs. □

6.7 SIMULATIONS

To test the algorithms, a variant of the model in Chapter 5 has been used.

6.7.1 Simulation Model

The simulation environment was described in Section 5.4. There are some differences between the model described there and the one used here. The main part is that the model used here only includes three nets and 21 agents. The number of inputs are only slightly higher than the number of inputs, and the transfer time is 0.01 times the cardinality compared to the model used in Section 5.4 where the transfer time was 0.005 times the cardinality.

6.7.2 Limitations

In Section 5.4 was described how the too complex models where removed. To be of any use, the algorithms in this chapter need at least two sets of local diagnoses. Therefore, the models where only one agent has found local diagnoses have been removed in the simulations.

6.7.3 Simulations

The algorithm in this Chapter starts when all local diagnose have been generated. For comparison, four different variations of the algorithm have been implemented. The first is the complete Algorithm 4 while the others are variations on this.

The second has replaced FindSubGraph with the more crude partition of the agents that was described in the beginning of Section 6.1. The third has replaced FindR with an ordering so that the agent with largest smallest diagnosis is first, and then the rest follows in descending order. The last uses the direct method described in the beginning of Section 6.1. This method does not keep any information in memory, instead all diagnoses with size lower than $\mathcal{L}$ is merged at each repetition.

6.7.4 Result

Mean values of the computation times can be seen in Figure 6.5. With this model, the number of modules found by FindSubGraph and the more simple partition are mostly the same. Therefore, only a small difference can be seen in the mean times, most notably at 900 and 1000 components.

The direct variant is slower than the other three for the simulations with 500 or more components. For 900 components, the max-first variant is faster, but the complete algorithm and the crude module-differentiation algorithm is faster in the other cases. The knee in the simulation times seen for 900 components is probably due to the fact that the system is chosen by random. To conclude, the complete algorithm is mostly faster than the other three variants.

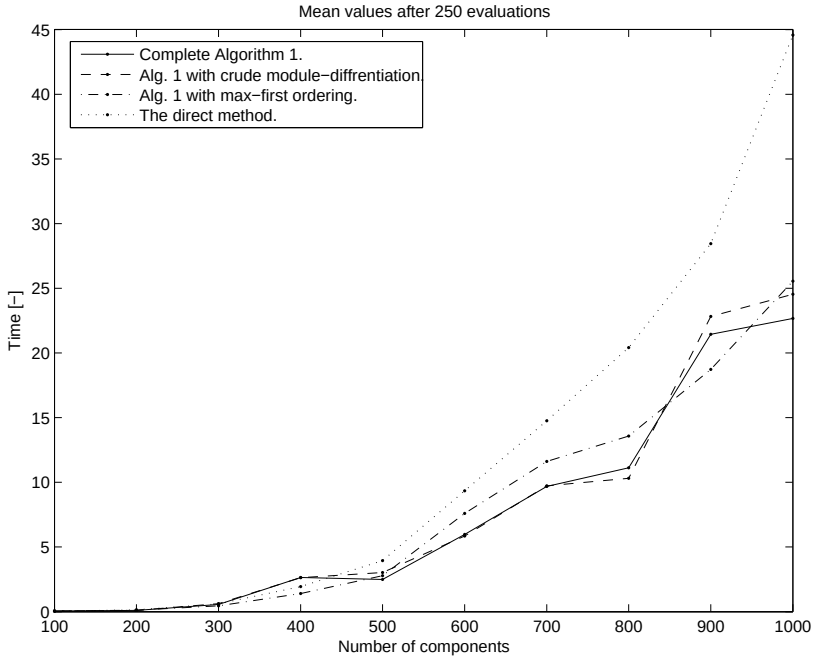


FIGURE 6.5: Computation times for the algorithms with increasing number of components.

7

CONCLUSIONS PART I

A framework for distributed diagnosis has been presented. The framework extends consistency based diagnosis to distributed systems. The system consists of agents that are connected to each other via a network. Each agent can state diagnoses for the local objects, which consists of both components and input signals. It was possible to use the framework to describe how faulty components affects other parts of the system.

The set of global diagnoses includes all diagnostic knowledge in the system. It would therefore be advantageous if these where available to the local agents. The simplest approach to do this is to merge all local diagnoses to obtain the global diagnoses. This approach might however be very computer intensive, therefore, this part of the thesis described how it was possible to use these local diagnoses to calculate more complete local diagnoses. After this, it was described how the more likely global diagnoses could be calculated.

To extend the diagnoses, conflicts could be transmitted between the agents. The choice of which conflicts to transmit was based on the conflicts themselves. Alternatively the local diagnoses could be used to make the choice. The gain when diagnoses were used was that only those conflicts that more probably would give better extended local diagnoses could be transmitted.

Simulations were used to show how algorithm for the sharing of conflicts behaved. Four different variants was simulated, the trans-

mission of conflicts where the choice of which conflicts to transmit was based on conflicts, and one where it was based on the local diagnoses. These two variants was then simulated with and without virtual components. Without loss of fault detection degree, it was possible to reduce both the simulation time and the number of extended local diagnoses.

Another way to extend the local diagnoses was to transmit local diagnoses. It was directly possible to only transmit those diagnoses with a higher probability to be correct.

A problem when transmitting diagnoses is that the size and the number of the diagnoses grow. One way to reduce this growth was as stated above to minimize the number of transmitted diagnoses. To further reduce the growth, it was shown how the size of the conflicts could be reduced while keeping the integrity of the diagnoses.

It has been shown how an algorithm that uses the agents' local diagnoses to calculate the minimal cardinality module diagnoses could be designed. The algorithm is distributed in such a manner that it could use the agents' own processing power, this reduces the need for a central diagnostic agent. Simulations was used to show how the complexity was reduced when different parts of the algorithm was used.

II

Simulation Based Residual Generators

8

Introduction Part II

When designing model-based fault-diagnostic systems, *residual generators* are often used. Each residual generator is designed such that the residual is different from zero when some fault is included in the supervised system. Different residual generators are sensitive to different sub sets of faults, and thereby isolation can be achieved. One approach to construct residual generators is to first find smaller sub sets of the model equations, such that each sub set is overdetermined. Since the sets are overdetermined, they can be used to construct residual generators.

This part of the thesis presents a new method for constructing residual generators for minimally overdetermined sets of equations, which means that there is exactly one more equation than unknown variables. The main idea is that by adding an extra variable, denoted the residual variable, an exactly determined system of equations is designed. The method is to simulate this set of equations, and thereby determine if the residual variable is zero or not. The method avoids the need to analytically transform the sets of *differential algebraic equations* (DAEs) into some specific residual generator form.

The set of equations with a residual variable could be used as a residual generator if the residual variable has been added to the set of equations such that the set of equations, in a practical sense, is *simulatable*. More specifically, this means that there are requirements on uniqueness, index, and stability. The general framework will be presented in Chapter 9.

Compared to using the whole model, there are at least two advantages to start with a minimally overdetermined set of equations. The first is that this set is typically small, which follows from the fact that it is minimal. A small model is normally easier to handle than a large model. The second advantage is that some decoupling of faults and disturbances has already been achieved, since only the equations that are necessary for the minimally overdetermined set are included in the set. The result is that the amount of decoupling that has to be done is reduced. Given a set of model equations, one approach to find the minimally overdetermined sets of equations is to find all *minimal structurally overdetermined* (MSO) sets of equations. The reason for using MSO sets is that these sets are the smallest sets of equations that can be used to form residual generators. Further on, it has been shown in [Kry03, KN02] that these sets characterize all the diagnosability properties of the system.

When extracting MSO sets, different assumptions about the structural relationship between the unknown variables and about derivatives of sensor values, will result in different sets of MSO sets. From these assumptions two main approaches are found, the *direct* and the *static* approach. The direct approach results in a set of equations that includes dynamic state variables, and in the general case the MSO set is a set of DAEs. The static approach results in a set of equations with only instantaneous state variables, which therefore results in a static system of equations. It is also possible to make a combination of the two approaches; this will be called the *partially static* approach.

An advantage of the direct approach is that no approximations of sensor derivatives have to be calculated. A disadvantage is that, sometimes, solution stability and complexity of the DAEs can be a problem. The benefit with the static approach is simplicity when constructing the residual generator, and the disadvantage is that derivatives of sensor values have to be approximated.

In Chapter 10, the method is demonstrated on a non-linear point-mass satellite system taken from [PI01]. The satellite system is non-linear and includes an unknown variable that has to be decoupled. The system consists of 7 equations and the residual generators typically consists of 3 to 4 equations. The example is used to illustrate how different residual generators could be designed depending on which derivatives of sensor values that are approximated.

Additionally, it is in Chapter 11 demonstrated how the method can take advantage of the modeling language Modelica. To demonstrate the method, a non-trivial part of a paper mill consisting of over 20 equations is used. The implemented residual generators typically consists of 7 to 9 equations. It is demonstrated how to extract the equations from the model, how to add a residual variable, and how to simulate the resulting residual generator.

8.1 OUTLINE OF PART II

The outline was given in the thesis introduction and are repeated here for completeness.

Chapter 8 Introduction Part II.

Chapter 9 In this chapter, a method for the construction of residual generators is described. The residual generators are constructed from smaller sub sets of model equations. These sets are overdetermined such that there exist exactly one more equation than unknown variables in the set of equations.

By adding a residual variable to the set of equations, an exactly determined set of equations is constructed. It is shown how the residual variable can be added such that the set of equations can be simulated. The method avoids the need to analytically transform the sets of equations into some specific residual generator form.

It is shown how it is possible to choose between either a more complex residual generator or the need to perform derivative approximations of sensor values.

Chapter 10 The method presented in the preceding chapter is exemplified on a model for a non-linear point-mass satellite. It is shown how the addition of approximations of derivatives of sensor values results in different residual generators.

Chapter 11 It is shown how the method can take advantage of the simulation tool Dymola which use the simulation language Modelica to extract and simulate the set of equations.

Chapter 12 Conclusions Part II.

8.2 RELATED WORK

The MSO property is discussed in [Kry03] and the shorter [KÅN05]. Structural analysis for diagnosis of a valve is used in [FDKC03]. All of these publications includes discussions about how the MSO sets can be extracted from a model and be used for predicting the structural diagnosability.

The paper [FÅ05] considers design of observers for sets of DAEs or descriptor models. The observers can be used as residual generators and is formulated as sets of DAEs.

The example in Chapter 10 is taken from [Rug96]. It is used in [PI01] for a solution of the same problem using geometric methods.

In Chapter 11, the simulation language Modelica and the simulation tool Dymola are used to perform simulations. The integration methods that can be used to simulate models in Dymola are presented in for example [Elm02] and [Fri04].

8.3 PUBLICATIONS

The results in chapter 9 and 10 have been presented in the following publications.

- [BN02] – This paper presented a first formulation of the material in Chapter 9.
- [BN03] – This is and extended and revived paper of the above publication. The paper presents the material in Chapter 9 and 10.

9

SIMULATION BASED RESIDUAL GENERATORS

In the preceding chapter a method for the design of residual generators was introduced. The method will be further described in this chapter. First, however, some properties for simulation tools will be described. This will be followed by a model description and a deeper discussion about the MSO property. How to calculate the MSO sets from a model is also discussed. The chapter further describes the constraints that need to be fulfilled when the residual variable is added to the MSO set of equations.

9.1 SIMULATION TOOLS

Numerous simulation tools exist for computing numerical approximations to the solution of a wide variety of differential equation systems. With a simulation tool is meant a graphical or text-based tool, such as Simulink, Dymola, gPROMS, MATLAB, etc. These are a user interfaces to the actual solvers, which are the software that simulates the model. Solvers primarily include standard ODE-solvers such as ODE15 in Simulink, and DAE-solvers such as DASSL [AP98] that is implemented in most simulation tools that handles DAEs.

When models are considered, an important property is the index of the model. For a set of equations, the index is often defined as the number of times a set of equations must be analytically differentiated

with respect to time, such that an explicit ODE can be formed. For a more exact definition see for example [CG95, AP98].

A limitation for most solvers is that they can only handle problems with index less than or equal to one. Even though there exist solvers such as RADAU5 that can solve some problem of index higher than one, these solvers can only solve some specific classes of problems with higher index. The method presented in this chapter will only use solvers for models with index less than or equal to one, because these solvers are the only one that can solve general non-linear DAEs.

9.2 SYSTEM MODEL

The following definition of model will be used both to describe the system that residual generators should be constructed for, and to describe the residual generators.

DEFINITION 9.1 (Model): *A model is a set of equations in differential algebraic form*

$$0 = G(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}, \mathcal{F})$$

where $\mathcal{X}$ is the set of unknown dynamic state variables, $\mathcal{Z}$ is the set of unknown instantaneous state variables, $\mathcal{U}$ is the set of known inputs and outputs, and $\mathcal{F}$ is the set of unknown faults. The set of sensor variables $\mathcal{Y}$ are included in $\mathcal{U}$, i.e. $\mathcal{Y} \subseteq \mathcal{U}$.

Each equation $e \in G(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}, \mathcal{F})$ is associated with assumptions

$$ass(e)$$

which includes limitations for the variables included in the equation.

The faults are assumed to be zero in the fault free case. All known variables have been included in the set $\mathcal{U}$ which includes actuator signals, sensor signals, and constant variables. The higher derivatives of the known signals are assumed known, *with the exception* of the sensor variables. The reason for this is that the derivatives of the sensor variables might be difficult to obtain.

EXAMPLE 9.1: An example of a model is

$$\begin{aligned}
e_1: \quad 0 &= -\dot{\rho} + \tfrac{u_1}{\rho} + f + \sigma, \quad ass(e_1) = \{\rho > 0\} \\
e_2: \quad 0 &= \sigma + u_2 \\
e_3: \quad 0 &= -y_1 + \rho + \sigma \\
e_4: \quad 0 &= -y_2 + \rho
\end{aligned}$$

where f is an additive fault. For this model, $G = \{e_1, e_2, e_3, e_4\}$, $\mathcal{X} = \{\rho\}$, $\mathcal{Z} = \{\sigma\}$, $\mathcal{U} = \{u_1, u_2, y_1, y_2\}$, and $\mathcal{F} = \{f\}$. Equations e_3 and e_4 include sensor variables, i.e. $\mathcal{Y} = \{y_1, y_2\}$. ◇

9.3 MSS SETS OF EQUATIONS

The main property of an MSO set is that there is *exactly one* more equation than unknown variables included in the equations. Another important property is that no proper subset of the MSO set is an MSO set. The definitions from [KÅN05], in a slightly different notation, are cited below.

DEFINITION 9.2 (Structurally overdetermined): *A finite set of equations* E *is* structurally overdetermined *with respect to the set of variables* X *in* E *if* $|E| > |X|$.

DEFINITION 9.3 (Minimal Structurally overdetermined): *A structurally singular set is a* minimal structurally overdetermined *(*MSO*) set if none of its proper subsets are structurally overdetermined.*

For more information see [Kry03, FDKC03, KÅN05]. From the equations in Definition 9.1 can an MSO set for the model be described.

A set of equations

$$g(\cdot) \subseteq G(\cdot)$$

is an MSO set for the model G if the set g is minimal structural overdetermined with respect to the variables $\dot{\mathcal{X}}$, $\mathcal{X}$, and $\mathcal{Z}$, where $\dot{\mathcal{X}}$ and $\mathcal{X}$ are considered to be the same variable.

Each MSO set is in itself a model in the form

$$0 = g(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}, \mathcal{F}).$$

From the model, a set of MSO sets $\{\text{MSO}^1, \ldots, \text{MSO}^k\}$ can be found. How these are found will be discussed later in Section 9.6. The problem now is to construct a residual generator for each MSO set of equations. This residual generator should be simulatable and preferable be sensitive to the faults included in the MSO set of equations.

EXAMPLE 9.2: Consider Example 9.1. An MSO set for the model defined in the example is the set of equations $\{e_1, e_2, e_4\}$. Notice that the MSO set includes three equations and two unknown variables, i.e. $\dot{\rho}$ and ρ, which are considered as one and the same variable, and σ. This MSO set includes information about the fault f. With a correctly added residual variable, the information about the faults size can be extracted.

Another MSO set for the model is the set of equations $\{e_2, e_3, e_4\}$. Notice that the MSO set includes three equations and two unknown variables, i.e. ρ and σ. ◇

9.4 RESIDUAL GENERATORS

For each MSO set found from the model, it is assumed that there exists initial values $\dot{\mathcal{X}}_o$, $\mathcal{X}_o$, and $\mathcal{Z}_o$ such that

$$0 = g(\dot{\mathcal{X}}_0, \mathcal{X}_o, \mathcal{Z}_0, \mathcal{U}, 0)$$

is satisfied.

The state variable and input data space is limited by the assumptions stated when the model is defined. These assumptions might include physical information, e.g. pressure is always greater than zero, but also information about when the equations are valid, e.g. control signals over some given value. For a given MSO set, the assumptions define a variable space.

DEFINITION 9.4 (Variable space): *Let* g *be a set of equations where each equation* $e \in g$ *includes some assumptions* $\mathtt{ass}(e)$. *The* variable space *is*

$$\Omega(g) = \{\{\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}\} | \{\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}\} \in \mathtt{ass}(e), e \in g\}.$$

The variable space includes knowledge about the system that can be used when constructing residual generators.

EXAMPLE 9.3: (Variable space) Consider Example 9.1, where e_1 includes the assumption $\rho > 0$. If this is the only assumptions then

$$\Omega(g) = \{\rho > 0, \dot{\rho}, \sigma, u_1\}$$

The unconstrained variables will be dropped when no misunderstanding is likely to occur, i.e.

$$\Omega(g) = \{\rho > 0\}$$

will be used. ◇

The main idea when constructing a residual generator in this approach is to add a *scalar residual variable*, $r \in \mathbb{R}^1$ and possibly some of its derivatives, to the MSO set. The residual should be added such that the MSO set with the additional residual variable can be simulated.

EXAMPLE 9.4: Consider the model

$$\begin{aligned} e_1: \quad 0 &= \dot{x} - x \\ e_2: \quad 0 &= y_1 - x + f \\ e_3: \quad 0 &= y_2 - x. \end{aligned}$$

Two MSO sets that includes the fault f are the set $\{e_1, e_2\}$ and the set $\{e_2, e_3\}$. These sets can be used to form residual generators. To the first can for example r be added such that

$$\begin{aligned} 0 &= \dot{x} - x + \alpha_1 r \\ 0 &= y_1 - x + \dot{r} + \alpha_2 r. \end{aligned}$$

In this residual generator the first derivative of the residual has been added. The parameters α_1 and α_2 can be used to stabilize the model.

The second MSO set can be used to form the residual generator

$$0 = y_1 - x + r$$
$$0 = y_2 - x.$$

This residual generator is static in contrast to the first that was dynamic. ◇

The MSO set with the additional residual variable is denoted an MSO model (slight misuse of the acronym MSO).

DEFINITION 9.5 (MSO model): *Given an* MSO *set of equations* g. *If the highest derivative of the residual variable* r *is* n *and the parameter matrix* $\Gamma \in \mathbb{R}^{|g|\times(n+1)}$, *then an* MSO model *for the set* g *is a model where the set of equations is*

$$0 = g(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}) + \Gamma R \tag{9.1}$$

and $R = [r, \dot{r}, \ldots, r^{(n)}]^T$.

How the residual variable and its derivatives are added is a freedom that can be used when designing the MSO model. Notice that the variables $\dot{\mathcal{X}}$, $\mathcal{X}$, $\mathcal{Z}$, and $\mathcal{U}$ are not the same as in the model that the MSO set was extracted from. The following example illustrates the definition of MSO model.

EXAMPLE 9.5: Consider the model in Example 9.1. An MSO set is $\{e_1, e_2, e_4\}$, and an MSO model for this set with $n = 1$ is

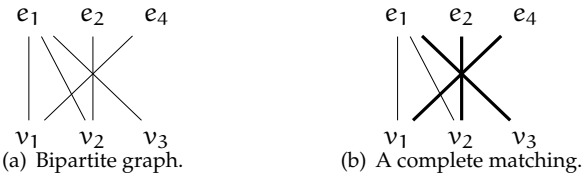
$$0 = \begin{bmatrix} \dot{\rho} - u_1 - \sigma \\ \sigma + u_2 \\ y_2 - \rho \end{bmatrix} + \Gamma \begin{bmatrix} r \\ \dot{r} \end{bmatrix}$$

where $\Gamma \in \mathbb{R}^{3\times 2}$. The dynamic variable is $\mathcal{X} = \rho$, the instantaneous variable is $\mathcal{Z} = \sigma$, and the known variables are $\mathcal{U} = [u_1, u_2, y_2]^T$. If fault f is sufficiently large and Γ is correctly designed, then simulations of this MSO model can be used to extract the residual r that is sensitive to fault f.

The other MSO set in Example 9.1 was the set of equations $\{e_2, e_3, e_4\}$. The MSO model for this set with $n = 1$ is

$$0 = \begin{bmatrix} \sigma + u_2 \\ y_1 + \rho + \sigma \\ y_2 - \rho \end{bmatrix} + \Gamma \begin{bmatrix} r \\ \dot{r} \end{bmatrix}$$

where $\Gamma \in \mathbb{R}^{3\times 2}$. The dynamic variable is $\mathcal{X} = \emptyset$, the instantaneous variable is $\mathcal{Z} = \{\sigma, \rho\}$, and the known variables are $\mathcal{U} = [u_2, y_1, y_2]^T$. ◇



(a) Bipartite graph. (b) A complete matching.

FIGURE 9.1: Bipartite graph and a complete matching of the bipartite graph.

When simulating an MSO model, the highest derivative of the residual variable $r^{(n)}$ is unknown, while the lower derivatives are known from integration of $r^{(n)}$. If this MSO model is simulatable then it can be used as a residual generator.

The residual variable can not be added to the MSO set arbitrary. Due to structural and analytical properties for the set, there are *constraints* that Γ must fulfill. The highest derivative must at least be added to one of the analytically redundant equations, this will be discussed more in the next section.

9.5 REDUNDANT EQUATIONS

First will a brief introduction to graphs be given, this is followed by a discussion about redundant equations.

9.5.1 Bipartite Matching

A *graph* G consists of vertices V (sometimes called nodes) and edges E, where each edge connects two vertices. A *bipartite graph* is a graph where the vertices are partitioned into two disjoint sets V_1 and V_2. A complete bipartite matching exists if there is some subset $\bar{E} \subseteq E$ that connects each $v_1 \in V_1$ to one unique $v_2 \in V_2$. For more information see for example [Har69].

EXAMPLE 9.6: Figure 9.1(a) shows a bipartite graph consisting of three equations and three variables, i.e. two sets of vertices, and edges connecting these vertices. A complete matching of the graph is shown in Figure 9.1(b). Equation e_1 is for example matched to variable v_3. ◇

9.5.2 Structurally and Analytically Redundant Equations

Two different types of equation redundancy is used here.

DEFINITION 9.6 (Structurally redundant): *Given a set of equations* g. *An equation* $e \in g$ *is* structurally redundant *if for a set of unknown variables*

$$X \subseteq \{\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}\},$$

there exist a complete bipartite matching between the unknown variables in $g \backslash e$ *and the equations* $g \backslash e$.

DEFINITION 9.7 (Analytically redundant): *An equation* $e \in g$ *is* analytically redundant *if* e *is structurally redundant for all*

$$X \in \Omega(g)$$

where X *is the unknown variables in* g *and* Ω *is the variable space.*

An equation is *analytically redundant* if it is structurally redundant for *all* operating points in Ω. Note that an equation is structurally redundant *if* it is analytically redundant.

When adding r to an MSO set, the constraints are that, firstly, $r^{(n)}$ must be added to at least one of the analytically redundant equations. Secondly, if it is added to several analytically redundant equations then Γ must fulfill some additional constraints, which are discussed later in Section 9.5.4.

In this section methods to find *all* structurally and analytically redundant equations will be presented. The structurally redundant equations can be found with *structural* or *analytical* methods, while the analytically redundant equations can only be guaranteed to be found with analytical methods.

9.5.3 Finding Structurally Redundant Equations

To find the structurally redundant equations, structural analysis can be used. The structural analysis finds a bipartite matching between equations and unknown variables [KN02, Har69].

In Figure 9.2, a graph representing equation (9.1) is shown. If a complete matching exists for (9.1), between g and the unknown variables in g, then the equation that matches $r^{(n)}$ is structurally redundant. To find *another* structurally redundant equation, let $\bar{\Gamma}_j = 0$ and find a new complete matching. Suitable repetition gives all structurally redundant equations.

EXAMPLE 9.7: Consider Example 9.1 and Example 9.2. The MSO set can be represented by the structural graph shown in the left part of Figure 9.3. The first row includes the equations and the second the unknown variables. The structural relationships are shown with lines. In the right part of the figure, a bipartite matching algorithm has been used to find a bipartite matching. The matching shows that e_4 is a structurally redundant equation.

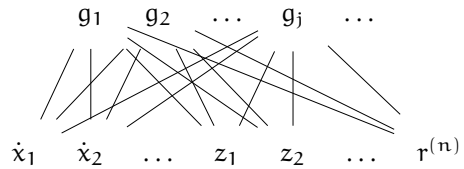
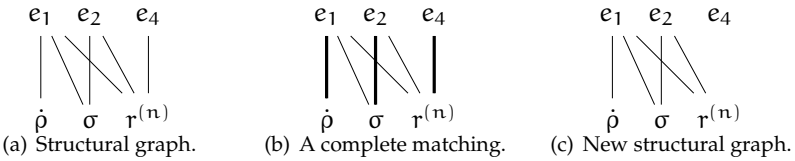


FIGURE 9.2: Bipartite graph representing (9.1). In the upper row are the equations, and in the lower row are all unknown variables included in the equations.



(a) Structural graph. (b) A complete matching. (c) New structural graph.

FIGURE 9.3: Structural model, complete matching of the model, and the model with the relation between the matched equation and $r^{(n)}$ removed.

Remove the relationship and find a new matching. In this example it is not possible to find any new bipartite matching, and the conclusion is that only equation e_4 is structurally redundant. ◇

9.5.4 *Finding Analytically Redundant Equations*

If some equations have been found to be structurally redundant, it should be checked if they are also analytically redundant.

To find the analytically redundant equations, the *implicit function theorem* will be used. To investigate if an equation is analytically redundant, it should fulfill the conditions in the theorem in all operation points, as defined by Ω. To test if a set of equations fulfills the conditions in the theorem, a common approach is to test if the *Jacobian* is non-singular [V$^+$95, MS92]. Equation (9.1) will only fulfill the theorem in all operation points *if* $r^{(n)}$ has been added to an analytically redundant equation.

This means that the Jacobian determinant

$$|J| = \left| \frac{\partial(g(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}, 0) + \Gamma R)}{\partial[\dot{\mathcal{X}}, \mathcal{Z}, r^{(n)}]^T} \right| \neq 0$$

where $\{\dot{\mathcal{X}}, \mathcal{Z}, r^{(n)}\}$ are the unknown variables. Due to the special construction of (9.1), the Jacobian determinant is

$$|J| = \xi^T(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U})\bar{\Gamma} \tag{9.2}$$

where $\xi(\cdot)$ is a vector and $\Gamma = [\Gamma_1, \ldots, \bar{\Gamma}]$, i.e. $\bar{\Gamma}$ is the column corresponding to $r^{(n)}$. The sum $\xi^T(\cdot)\bar{\Gamma}$ arises from the definition of determinant, i.e. sum of vector variable times co-factor.

The models used in this part are quite specific since the objective is to find the analytically redundant equations. It is therefore possible to consider only those $\bar{\Gamma}$ that only include one equation that is nonzero. Let $\bar{\Gamma} = [0, \ldots, \bar{\Gamma}_j, \ldots, 0]^T$, where $\bar{\Gamma}_j \neq 0$. If $|J| \neq 0$ then the j:th equation is structurally redundant. To decide if the j:th equation is analytically redundant, the variable space Ω has to be considered.

PROPOSITION 9.1: *Given an* MSO *model which includes the equations* g *and the parameter matrix* Γ*. If there does not exist*

$$(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}) \in \Omega(g)$$

such that

$$\xi_j(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U}) \in 0$$

where $\xi(\cdot)$ *is given by (9.2), then the* j*:th equation is guaranteed to be analytically redundant.*

The meaning is that, if it is not theoretically possible for ξ_j to equal zero, then the j:th equation is analytically redundant.

9.6 APPROACHES TO EXTRACT MSO SETS

When the MSO sets are extracted, different assumptions about which derivatives of sensor values that are available, will result in different sets of MSO sets, and consequently in different residual generators.

9.6.1 Finding MSO *Sets given a Structural Model*

In [KN02] a systematic and automatic algorithm is presented, called the MSO-algorithm. The algorithm finds the set of MSO sets for a given structural model. This algorithm was further improved in [Axe04] and it is this implementation that has been used in the examples to extract the plots of the structural relationships.

The algorithm is based on graph theoretical reasoning about the structure of the model, and is capable of handling non-linear differential algebraic non-causal equations. Further, the method is not limited to any special type of fault model.

The input to the algorithm is a *structural model* and information about which *derivatives of sensor values* that can be approximated. The structural model includes information about the connection between unknown variables and equations. By analyzing and manipulating

this structural model, the algorithm finds all MSO sets. In the algorithm, the variables $\dot{\mathcal{X}}$ and $\mathcal{X}$ are considered to be one variable.

The output of the algorithm is a set of MSO sets where each MSO set might include *differentiated* and *non-differentiated* equations from the original model.

9.6.2 Direct Approach

The direct approach to extract MSO sets is to directly find the MSO sets without considering *derivatives* of *sensor equations*. This means that *no* derivatives of sensor values are included in the MSO sets, e.g. sensor value y_1 is known while its derivative $\dot{y}_1$ is unknown. Note that differentiated non-sensor equations might be included in the MSO sets.

EXAMPLE 9.8: To continue Example 9.1 above. The MSO set $\{e_1, e_2, e_4\}$ is found with the direct approach. In this set, the dynamic state variable $\mathcal{X} = \{\rho\}$, the instantaneous state variable $\mathcal{Z} = \{\sigma\}$ and the known variables $\mathcal{U} = \{u_1, u_2, y_2\}$. ◇

The MSO-algorithm is given the structural model, where $\mathcal{X}$ and $\mathcal{Z}$ are considered unknown. The output is a set of MSO sets. The sets can include both differentiated and non-differentiated equations. Further, since the MSO model sometimes is a non-linear DAE, a non-linear DAE solver is preferably used to simulate the MSO model.

A disadvantage with this approach is that, sometimes, solution stability and complexity due to models with index higher than zero can be a problem.

9.6.3 Using Derivative Approximations

In some cases, the direct approach does not generate a set of MSO sets with acceptable detection and isolation properties, or it is difficult to construct stable and fault-sensitive residual generators. In these cases, it might be useful to find more MSO sets by adding approximations of sensor derivatives to the model, e.g.

$$y_{new} = \dot{y}. \tag{9.3}$$

One set of such MSO sets, that in the general case can not be implement, are those that have an index higher than one. In this case, it is often possible to include derivative approximations of sensor signals to the model, so that a new search, with the derivative sensor equation added, will find a corresponding MSO set with index zero or one. The new MSO set will be based on the same equations as the original MSO set, but it will in addition include differentiated sensor equations.

If all derivatives, i.e. $\{\dot{y}, \ddot{y}, \ldots\}$, of the sensor-equations included in the original MSO set are included in the new model, then an MSO set with index zero will be found.

If adding equations such as (9.3), then the structural relationship between a state variable and its derivative will from a simulation perspective not always exist. To reduce the risk of confusion, let $x_D \triangleq \dot{x}$ when the structural relationship between x and $\dot{x}$ has been removed in an MSO set. The following example illustrates this.

EXAMPLE 9.9: Consider the system

$$\begin{array}{lrcl} e_1: & 0 & = & -\dot{\rho} + \sigma \\ e_2: & 0 & = & -\dot{\omega} + \sigma \\ e_3: & 0 & = & -y_1 + \rho \\ e_4: & 0 & = & -y_2 + \omega + f \\ e_5: & y_{new} & = & \dot{y}_1 \quad \rightarrow \quad 0 = -y_{new} + \dot{\rho}. \end{array}$$

Two of the MSO sets that can be found for this system are $\{e_1, e_2, e_3, e_4\}$ and $\{e_1, e_2, e_4, e_5\}$, where the unknown variables are $\mathcal{X} = \{\rho, \omega\}$ and $\mathcal{Z} = \{\sigma\}$.

The first MSO set is

$$\begin{array}{lrcl} e_1: & 0 & = & -\dot{\rho} + \sigma \\ e_2: & 0 & = & -\dot{\omega} + \sigma \\ e_3: & 0 & = & -y_1 + \rho \\ e_4: & 0 & = & -y_2 + \omega \end{array}$$

An MSO model constructed from this set will have the dynamic variables $\mathcal{X} = \{\rho, \omega\}$ and instantaneous variables $\mathcal{Z} = \{\sigma\}$.

The second MSO set is

$$\begin{array}{lrcl} e_1: & 0 & = & -\rho_D + \sigma \\ e_2: & 0 & = & -\dot{\omega} + \sigma \\ e_4: & 0 & = & -y_2 + \omega \\ e_5: & y_{new} & = & \dot{y}_1 \quad \rightarrow \quad 0 = -y_{new} + \rho_D. \end{array}$$

When this system is simulated, ρ is decided from y_1 and $\dot{\rho}$ is decided from $\dot{y}_1$. Therefore, $\rho_D = \dot{\rho}$ and the model's dynamic variable is $\mathcal{X} = \{\omega\}$ and instantaneous variables are $\mathcal{Z} = \{\rho_D, \sigma\}$. ◇

9.6.4 *Static Approach*

If all derivatives of dynamic state variables in an MSO model are approximated by some old or new sensor equation, then the MSO model is static. This means that no dynamic state variables are included in the model, only instantaneous. If the MSO model is static then the set of equations can be solved with a general equation solver.

The benefit with the static approach is simplicity when constructing the residual generator. It also reduces the need for a dynamic solver. The disadvantage is that, often, derivatives of sensor values have to be approximated. In the cases where it is easy to obtain correct values of the derivatives, this approach is preferable used.

EXAMPLE 9.10: To continue the example above. With the static approach the MSO set $\{e_1, e_2, e_4, \dot{e}_4\}$ is found, where equation $\dot{e}_4$ is the time derivative of equation e_4. In this set, the dynamic state variable $\mathcal{X} = \emptyset$, the instantaneous state variables $\mathcal{Z} = \{\sigma, \rho_D, \rho\}$, and the known inputs $\mathcal{U} = \{u_1, u_2, y_2, y_2^D\}$. ◇

9.6.5 *Partially Static Approach*

The direct and static approaches are two extremes. It is possible to make a combination of the two approaches. If a subset of derivatives of sensor values can be approximated then the MSO models will be dynamic models with some dynamic states removed.

The main benefit is when the MSO models have a high index. By careful selection of which sensor values to approximate, a new MSO set with the same sets of equations but including some derivatives of the equations can be found, and thereby the index can be reduced.

9.7 SOME COMMENTS ON REDUNDANCY

It is possible that an equation is redundant for some variable subspace while being non-redundant for the remaining sub-space. The redundant equation vector $\bar{\Gamma}$ such that the Jacobian is guaranteed to be non-singular during the complete simulation. It is possible that for some $(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U})$ the Jacobian is singular. This singularity is induced because $\xi^T(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U})\bar{\Gamma} = 0$ for some space $(\dot{\mathcal{X}}, \mathcal{X}, \mathcal{Z}, \mathcal{U})$. The vector $\bar{\Gamma}$ should be designed such that this space is avoided.

9.8 SOME COMMENTS ON STABILITY

From Section 9.5 some constraints on Γ have been stated. The problem is to find the additional constraints on Γ that guarantees stability and give the residual good fault sensitivity. Depending on if the MSO model is linear, bilinear, non-linear, etc., different methods can be used to find the constraints. Note that the order of the highest derivative of r can be chosen to simplify the stability analysis. For a deeper study of the stability problem, see for example [FÅ05]. The stability problem will not be further studied in this thesis.

10

RESIDUAL GENERATORS FOR A SATELLITE MODEL

Two examples will be studied in this and the following chapter. The first is an example of a non-linear point-mass satellite; while the second is a part of a paper mill.

The first example is mainly used to illustrate the method described in Chapter 9, for example how approximations of derivatives of sensor values results in different sets of residual generators. The second example's main point is to demonstrate how the method can take advantage of the modeling language Modelica and the simulation tool Dymola to extract and simulate the residual generators.

10.1 INTRODUCTION

This example is an ideal model of a non-linear point-mass satellite system. It is taken from [Rug96, PI01]. Firstly the model is presented in Section 10.2. After this, the direct, static, and partially static approachs are considered in Section 10.4, 10.5, and 10.6 respectively. Finally, some simulation results are presented in Section 10.7.

10.2 PHYSICAL MODEL

The equations G describing the model are

$$\begin{aligned}
e_1: &\quad \dot{\rho} &=& \upsilon \\
e_2: &\quad \dot{\upsilon} &=& \rho\omega^2 - \theta_1 \tfrac{1}{\rho^2} + \theta_2 u_1 + d \\
e_3: &\quad \dot{\varphi} &=& \omega \\
e_4: &\quad \dot{\omega} &=& -\tfrac{2\upsilon\omega}{\rho} + \theta_2 \left(\tfrac{u_2}{\rho} + \tfrac{f_{u_2}}{\rho} \right) \\
e_5: &\quad 0 &=& -y_1 + \rho \\
e_6: &\quad 0 &=& -y_2 + \varphi + f_\varphi \\
e_7: &\quad 0 &=& -y_3 + \omega
\end{aligned}$$

where ρ and υ are radius and radial speed, φ and ω are angle and angular speed, u_1 is radial and u_2 tangential thrust, y_1, y_2, and y_3 are sensor signals, d unknown disturbance, f_{u_2} and f_φ bias fault in u_2 and y_2 respectively, and θ_1 and θ_2 are known constants. Equations $\{e_1, e_2, e_3, e_4\}$ defines the physical model and $\{e_5, e_6, e_7\}$ the sensor relations.

The equations includes the following assumptions

$$\begin{aligned}
ass(e_2) &= \{\rho > 0, \omega > 0, |u_1| \leqslant 1\} \\
ass(e_3) &= \{\omega > 0\} \\
ass(e_4) &= \{\rho > 0, \omega > 0, |u_2| \leqslant 1\} \\
ass(e_5) &= \{\rho > 0\} \\
ass(e_7) &= \{\omega > 0\}.
\end{aligned}$$

The variable space is therefore

$$\Omega(G) = \{\rho > 0, \omega > 0, \upsilon, \varphi, |u_1| \leqslant 1, |u_2| \leqslant 1, y_1, y_3, y_2\}.$$

The variable space limits the values for the state variables to positive radius and positive angular speed, etc. The actuator fault is limited to $|f_{u_2}| \leqslant 1$. Notice that sensor values y_1, y_2, and y_3 are included in $\mathcal{U}$, since these are input data to the MSO models. There are no simulation problems with the model.

10.3 STRUCTURAL MODEL

The structural description of the model is shown in Figure 10.1. The variables are on the x-axis, and the equations are on the y-axis. A circle $\circ$ means that the corresponding variable is included linearly, and a cross $\times$ means that it is included non-linearly. Time derivative is represented by $'$, e.g. $\rho' = \frac{d\rho}{dt}$.

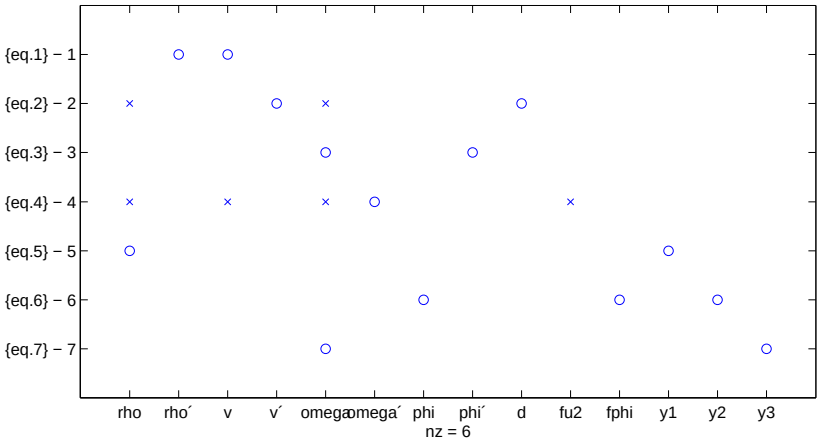


FIGURE 10.1: Structural model for the satellite example.

10.4 DIRECT APPROACH

In the direct approach, the MSO-algorithm is given the structural model and the information that no derivatives of sensor values can be approximated; see Section 9.6.2.

The MSO sets found by the MSO-algorithm are

MSO	Equation set	f_{u_2}	f_φ	d
MSO^1	$\{e_3, e_6, e_7\}$	0	X	0
MSO^2	$\{e_1, e_4, e_5, e_7\}$	X	0	0
MSO^3	$\{e_1, e_3, e_4, e_5, e_6\}$	X	X	0

An X in position i, j in the incidence matrix, which is the rightmost part of the table, means that set MSO^i might be sensitive to fault j. From the incidence matrix it can be concluded that it is ideally possible to detect and isolate both faults.

10.4.1 Structurally Redundant Equations

A structural analysis shows that in the set MSO^1, the equation e_6 is structurally redundant. The complete matching is shown in Figure 10.2.

It is, however, not possible to find a bipartite matching for the set MSO^2 and the set MSO^3. A result of this is that it is not possible to transform the MSO models to index zero without differentiation of sensor equations. The reason for the high index is that it is not possible to calculate v from the equations. One solution to reduce the index is to use an MSO model that includes equation e_2, if this was the case, then v could be integrated from $\dot{v}$.

(a) Complete matching. (b) New structural graph.

FIGURE 10.2: Complete matching and a new structural graph.

10.4.2 *Analytically Redundant Equations*

The set of dynamic state variables $\dot{x}$ is for set MSO^1, $\dot{x}^1 = \{\dot{\varphi}\}$. The set of instantaneous state variables z is $z^1 = \{\omega\}$.

The Jacobian for the set MSO^1 is

$$J^1 = \left[\begin{array}{cc|c} -1 & 1 & \\ 0 & 0 & \bar{\Gamma}^1 \\ 0 & 1 & \end{array}\right].$$

The Jacobian determinant is

$$|J^1| = \xi\bar{\Gamma} = [0, 1, 0]\,\bar{\Gamma}$$

and since $\xi_2 = 1$, with Proposition 9.1 it is concluded that equation 2 is analytically redundant.

10.4.3 *Design of* Γ *for set* MSO^1

A design example for Γ in set MSO^1 will here be presented. A first *design choice* is to add the residual and its first derivative to the MSO set. A second design choice is to let the parameters $\Gamma_{12} = \Gamma_{31} = \Gamma_{32} = 0$. This give the MSO model

$$0 = \begin{bmatrix} -\dot{\varphi} + \omega \\ -y_2 + \varphi \\ -y_3 + \omega \end{bmatrix} + \begin{bmatrix} \Gamma_{11} & 0 \\ \Gamma_{21} & \Gamma_{22} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix}. \tag{10.1}$$

In this case the model is linear and therefore, linear analysis will be used to find the constraints for Γ_{11}, Γ_{21}, and Γ_{22}. Transforming (10.1) to the frequency domain results in

$$r = \frac{y_2 s - y_3}{s^2\Gamma_{22} + s\Gamma_{21} + \Gamma_{11}}$$

which is stable if for example $\Gamma_{11} > 0$, $\Gamma_{21} > 0$, and $\Gamma_{22} > 0$. The model is stable with this design choice.

Notice that even though linear theory was used to find the constraints on Γ, the methods presented in Chapter 9 can not only be used for the linear case. In the general non-linear case other methods for stability analysis, for example Lyaponov theory, have to be used.

10.5 STATIC APPROACH

In the static approach the derivatives, up to some order, of all sensor values have to be approximated. The values are approximated using some software algorithm. Here only the first derivative will be used. The MSO-algorithm is given the information that all sensor derivatives can be approximated and finds *three* different MSO sets. The set of MSO sets found is

MSO	Equation set	f_{u_2}	f_φ	d
$\text{MSO}^{\tilde{1}}$	$\{e_3, \dot{e}_6, e_7\}$	0	X	0
$\text{MSO}^{\tilde{2}}$	$\{e_1, e_4, e_5, \dot{e}_5, e_7, \dot{e}_7\}$	X	0	0
$\text{MSO}^{\tilde{3}}$	$\{e_1, e_3, e_4, e_5, \dot{e}_5, \dot{e}_6, \dot{e}_7\}$	X	X	0

Time differentiation of an equation, e.g. $\dot{e}_6$, is performed analytically. For example, the $\text{MSO}^{\tilde{1}}$ set of equations is the set $\{\varphi_D = \omega, 0 = -y_{2D} + \varphi_D, 0 = -y_3 + \omega\}$.

10.5.1 Structurally Redundant Equations

For the three MSO models, the existence of bipartite matchings show that all equations in all MSO sets are structurally redundant.

10.5.2 Analytically Redundant Equations

For the three MSO models, the dynamic state variables $\dot{x} = \emptyset$ and $x = \emptyset$. The set of unknown instantaneous variables is $z^{\tilde{1}} = \{\varphi_D, \omega\}$, $z^{\tilde{2}} = \{\rho_D, \omega_D, \rho, \omega, v\}$, and $z^{\tilde{3}} = \{\rho_D,\ \varphi_D,\ \omega_D, \rho, \omega, v\}$. The Jacobians for the three MSO models are

$$J_{\tilde{1}} = \left[\begin{array}{cc|c} -1 & 1 & \\ 1 & 0 & \bar{\Gamma}^{\tilde{1}} \\ 0 & 1 & \end{array}\right] \qquad J_{\tilde{2}} = \left[\begin{array}{ccccc|c} -1 & 0 & 0 & 0 & 1 & \\ 0 & -1 & \alpha & -\beta & \lambda & \\ 0 & 0 & 1 & 0 & 0 & \bar{\Gamma}^{\tilde{2}} \\ 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & \\ 0 & 1 & 0 & 0 & 0 & \end{array}\right]$$

$$J_{\tilde{3}} = \left[\begin{array}{cccccc|c} -1 & 0 & 0 & 0 & 0 & 1 & \\ 0 & -1 & 0 & 0 & 1 & 0 & \\ 0 & 0 & -1 & \alpha & -\beta & \lambda & \\ 0 & 0 & 0 & 1 & 0 & 0 & \bar{\Gamma}^{\tilde{3}} \\ 1 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & \end{array}\right]$$

where $\alpha = (2v\omega - \theta_2 u_2))/\rho^2$, $\beta = 2v/\rho$ and $\lambda = 2\omega/\rho$.

The Jacobian determinants are for the three MSO models determined by ξ, where ξ for the three models are

$$\xi^{\bar{1}} = [1, 1, -1]^T$$
$$\xi^{\bar{2}} = [-\lambda, -1, 1, \alpha, -\lambda, -\beta, -1]^T$$
$$\xi^{\bar{3}} = [\lambda, \beta, 1, \alpha, \lambda, \beta, 1]^T.$$

Proposition 9.1 can be used to find out which equations that are analytically redundant in the MSO models. A closer examination of the set $\text{MSO}^{\bar{3}}$ shows that equations 3 and 7 are analytically redundant. To check if the remaining equations are analytically redundant, calculate the sets and check if they are empty,

$$\{(\rho, \omega, v, \varphi) | \xi_1^{\bar{3}} = \xi_5^{\bar{3}} = \frac{2\omega}{\rho} \in 0, (\rho, \omega, v, \varphi) \in \Omega(G)\} = \emptyset$$
$$\{(\rho, \omega, v, \varphi) | \xi_2^{\bar{3}} = \xi_6^{\bar{3}} = \frac{2v}{\rho} \in 0, (\rho, \omega, v, \varphi) \in \Omega(G)\} \neq \emptyset$$
$$\{(\rho, \omega, v, \varphi) | \xi_4^{\bar{3}} = \frac{2v\omega - \theta_2 u_2}{\rho^2} \in 0, (\rho, \omega, v, \varphi) \in \Omega(G)\}) \neq \emptyset.$$

From this it can be concluded that equation 2, 4, and 6 are not analytically redundant, while 1 and 5 are analytically redundant. Corresponding calculations can be carried out for the other MSO models.

10.6 PARTIALLY STATIC APPROACH

In Section 10.4 it was concluded that the MSO^2 model and the MSO^3 model does not include any redundant equations. The problem is that it is not possible to directly calculate v from the set of equations. A solution is to assume that it is possible to approximate the derivatives for some the sensors.

It is in this section assumed that $\dot{y}_1$ can be approximated. The two MSO sets that corresponds to MSO^2 and MSO^3 in Section 10.4 are shown below.

MSO	Equation set	f_{u_2}	f_φ	d
$\text{MSO}^{\hat{2}}$	$\{e_1, e_4, e_5, \dot{e}_5, e_7\}$	X	0	0
$\text{MSO}^{\hat{3}}$	$\{e_1, e_3, e_4, e_5, \dot{e}_5, e_6\}$	X	X	0

The difference is that the differentiated equation $\dot{e}_5$ is now included in the MSO models.

10.6.1 Analytically Redundant Equations

For the set $\text{MSO}^{\hat{2}}$ the dynamic state variable $\dot{x}^{\hat{2}} = \{\dot{\omega}\}$, and the instantaneous state variable $z^{\hat{2}} = \{\rho, \rho_D, v\}$. For the set $\text{MSO}^{\hat{3}}$ are $\dot{x}^{\hat{3}} = \{\dot{\varphi}, \dot{\omega}\}$ and $z^{\hat{3}} = \{\rho, \rho_D, v\}$.

The Jacobian determinants are

$$|J^{\hat{2}}| = [0, 0, 0, 0, 1]\,\bar{\Gamma}^{\hat{2}}$$
$$|J^{\hat{3}}| = [0, 0, 0, 0, 0, 1]\,\bar{\Gamma}^{\hat{3}}.$$

This means that for MSO$^{\hat{2}}$ model and MSO$^{\hat{3}}$ model, equations 5 and 6 are analytically redundant, respectively.

10.6.2 *Design of* Γ *for set* MSO$^{\hat{2}}$

In Section 10.6.1 it was concluded that equation e_7 in the MSO model is analytically redundant. A residual generator can be designed with the highest derivative $n = 0$, and parameters $\Gamma_{11} = \Gamma_{31} = \Gamma_{41} = 0$ and $\Gamma_{51} = 1$. This results in

$$0 = \begin{bmatrix} -\rho^D + v \\ -\dot{\omega} - 2v\omega/\rho + \theta_2 u_2/\rho \\ -y_1 + \rho \\ -y_1^D + \rho^D \\ -y_3 + \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \Gamma_{21} & \Gamma_{22} \\ 0 & 0 \\ 0 & 0 \\ \Gamma_{51} & \Gamma_{52} \end{bmatrix} \begin{bmatrix} \dot{r} \\ r \end{bmatrix}.$$

Let $\Gamma_{21} = K\Gamma_{51}$ and $\Gamma_{22} = K\Gamma_{52}$. After some manipulation of the model it can be seen that

$$\dot{\omega} = -(\frac{2y_1^D}{y_1} + K)\omega + \frac{\theta_2}{y_1}u_2 + Ky_3$$
$$\Gamma_{51}\dot{r} + \Gamma_{52}r = y_3 - \omega.$$

The model is locally stable if for example $\Gamma_{51} > 0$, $\Gamma_{52} > 0$, and $2y_1^D/y_1 + K > 0$. A design choice is $K = -2y_1^D/y_1 + K_1$ where $K_1 > 0$.

10.7 SIMULATIONS

To test the functionality of the MSO models that was analyzed in this chapter, two MSO models have been implemented.

10.7.1 *Residual Generators*

The MSO^1 model is chosen from Section 10.4 and the MSO$^{\hat{2}}$ model is chosen from Section 10.6. These MSO models are chosen because they illustrate two different approaches. The sensor derivative needed in MSO$^{\hat{2}}$ model is approximated with a discrete filter. MSO^1 model is sensitive to fault f_φ and MSO$^{\hat{2}}$ model to fault f_{u_2}. The influence structure

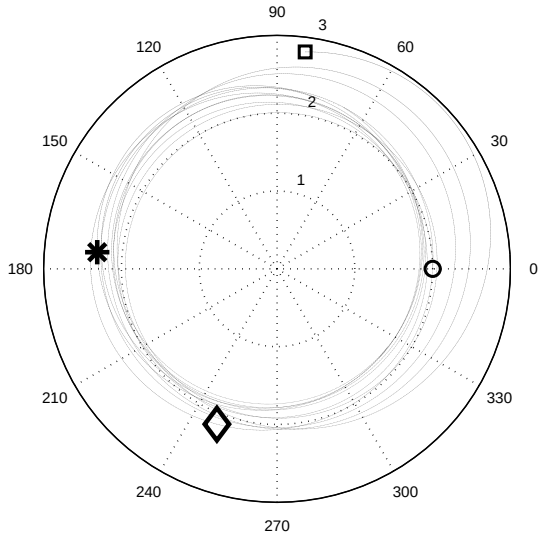


FIGURE 10.3: The satellite's movement in the polar plane. Start position is marked with $\circ$, start of fault f_φ is marked with $\star$, fault f_{u_2} is marked with $\diamond$, and end position is marked with $\square$.

is shown below.

MSO	f_{u_2}	f_φ	d
MSO^1	0	X	0
$MSO^{\hat{2}}$	X	0	0

Full detectability and isolability are achieved.

The chosen MSO models are stable. MSO^1 model has parameters $\Gamma_{11} = 0.2$, $\Gamma_{21} = 1$ and $\Gamma_{22} = 1$. $MSO^{\hat{2}}$ model has parameters $K_1 = 0.1$, $\Gamma_{51} = 1/2$, and $\Gamma_{52} = 1/4$. The residuals have been scaled such that the probability for false alarm is 0.04 % in the fault-free case. The threshold is thereby set to ± 1.

10.7.2 Simulation

The model and the two MSO models have been implemented in Simulink [Mat05]. The disturbance d is implemented as band limited white noise. Figure 10.3 shows the satellite's movement in the polar plane. The Start position is marked with a circle ($\circ$), the start of fault f_φ with a star ($\star$), the start of fault f_{u_2} with a diamond ($\diamond$), and the end position with a square ($\square$). The satellite first moves in approximately the same track, and then when the engines are used the satellite increases its altitude.

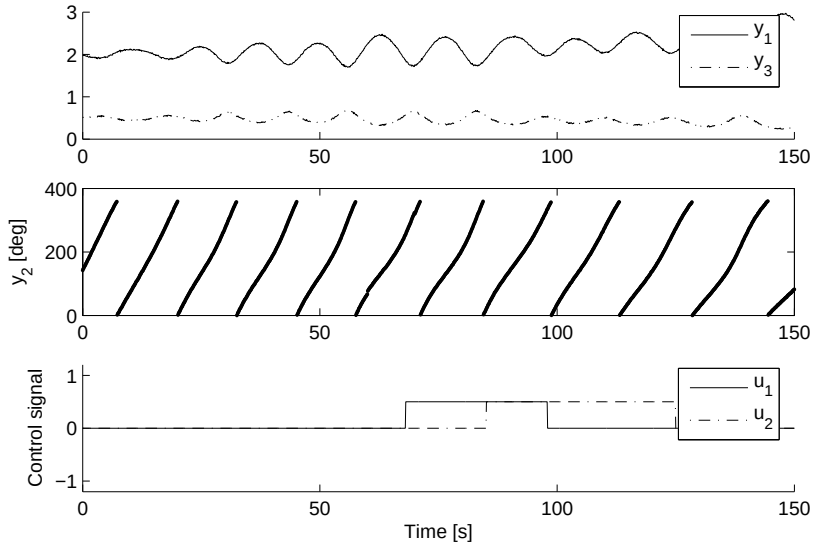


FIGURE 10.4: Sensor values and control signals.

Figure 10.4 shows the sensor values and the control signals. Radial thrust is applied at time 68 s to 98 s, and tangential thrust is applied at time 85 s to 125 s. Figure 10.5 shows the applied faults. Fault f_φ is applied at time 60 s with a size of 7.5° and removed at time 70 s. Fault f_{u_2} is introduced at time 90 s with a size of -0.4 and removed at time 125 s. The actuator fault represents a complete loss of tangential thrust.

10.7.3 Result

Figure 10.6 shows the residual values and the corresponding logic decision. Fault f_φ is *detected* at time 60.1 s and f_{u_2} at 98.3 s. As predicted, the residual generator based on MSO^1 reacts to the introduction of fault f_φ and the residual generator based on MSO^2 reacts to fault f_{u_2}. The first residual generator reacts weakly on the sensor fault, this can be seen in the initial growth of the residual, and then the gradual return to zero. Considering the satellite model, this result is not surprising. Notice also that the residual detects the removal of the sensor fault. The second residual generator reacts strongly to the actuator fault.

The conclusion is that both MSO models react to correct fault and that *detection and isolation are performed correct*. The simulation results show that the two MSO models that have been implemented react correctly to the introduced faults.

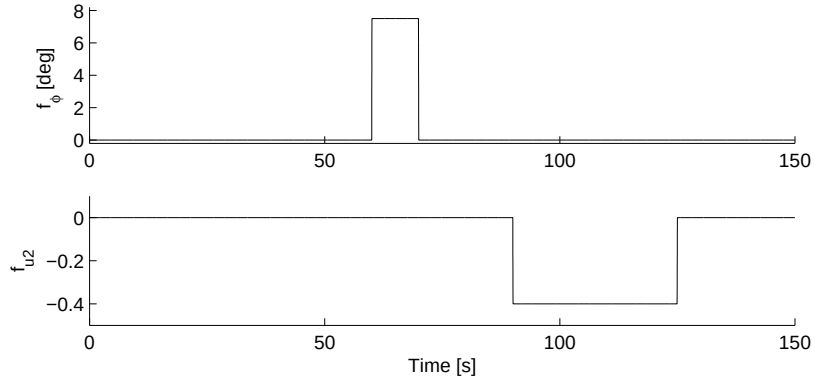


FIGURE 10.5: Fault signals.

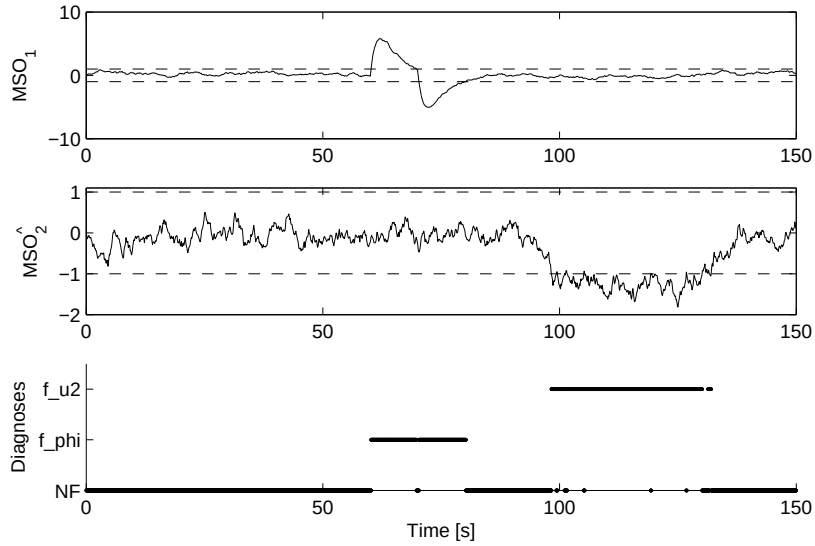


FIGURE 10.6: Residual values. The dashed lines are the thresholds. In the bottom is the logical conclusion from the residuals' values shown.

11

SIMULATIONS USING MODELICA

Residual generators are constructed and evaluated with the help of the modeling language Modelica and the simulation tool Dymola. As an experimental system, a part of the stock preparation and broke treatment system in a paper mill is used. Four different residual generators are implemented and analyzed in this chapter. The objective is to demonstrate how the method can take advantage of the modeling language Modelica and the simulation tool Dymola to extract and simulate the residual generators.

11.1 MODELICA

Modelica is an object-oriented language for modeling of physical systems for simulation purposes [Fri04]. The language is non-causal in its implementation and these non-causal properties are used in this chapter. The simulation tool Dymola [Elm02] is used to perform the actual simulations. The tool handles models defined in the Modelica standard. Several solvers are implemented in Dymola, of which the DASSL solver will be used in all the simulations in this chapter.

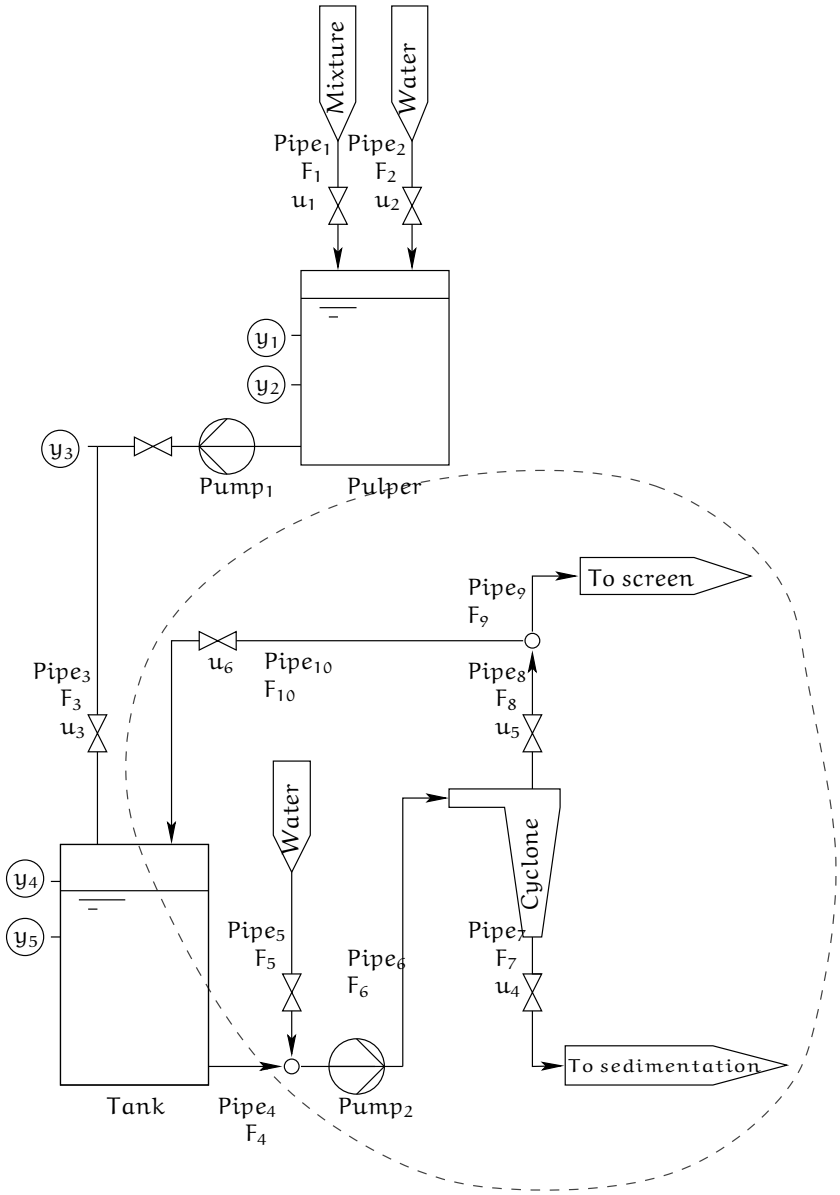


FIGURE 11.1: Schematic picture of the stock preparation and broke treatment system. Flows are denoted by F, control signals by u, and sensors by y.

11.2 MODEL BACKGROUND

A first model of the system was developed by ABB Corporate Research. One of the objectives was to use the model for model-based diagnosis. The original model was however not suitable for this objective, and it was therefore greatly simplified and presented in its new form in [Bit01]. The model was analyzed with structural methods in [Kry03, KN02].

11.3 STOCK PREPARATION AND BROKE TREATMENT MODEL

The system used for the stock preparation and broke treatment part of the paper mill is schematically shown in Figure 11.1. The system is used to prepare paper mixture for use in the paper machines. The system starts with recycled paper and water. The recycled paper is simulated as a fluid with a high concentration of paper fibers; it is the mixture flow in the figure. The two parts are mixed in the pulper tank and when the concentration is right, the fluid is moved thru pipes to a tank by using a pump. From this tank, the mixture is pumped to a cyclone. The cyclone cleans the mixture, which is then sent to the screening process. The model is described by a non-linear DAE that will be described in this section.

11.3.1 Limitations

The intention with this project was not to construct a complete diagnostic system for the model. Therefore, only a part of the system was thoroughly analyzed.

The system studied here is limited to the pulper, the tank, pipe one to four, pump one, and five sensors. The sensors measure the level and the concentration in the pulper, the outflow from the pulper, and further the level and change of the level in the tank.

11.3.2 Variable Definitions

The variable definitions are shown in Table 11.1 and the type definitions used in the model are shown in Table 11.2. Note that many of the types are derived from the standard Modelica library `Modelica.SIunits`.

The specific constraints on the variables are also shown in the table. Notable is that the concentration is limited to $[0, 1]$. These constraints

TABLE 11.1: Variable definitions.

Description	Variable	Note
Dynamic state	L_i, χ	$i \in \{1,2\}$
Instantaneous state	δ_{cv_i}, δ_{fric_i}, δ_{pump}, p, F_i, F_3^{air}, F_3^{fluid}	$i \in \{1,2,3\}$
Constant	p_{atm}, p_j, a_i, b_i, d_j, g, ρ, A_j, χ_j, F_4	$i \in \{1,2,3\}$, $j \in \{1,2\}$
Control signal	u_i	$i \in \{1,2,3\}$
Sensor signal	y_i	$i \in \{1,2,3,4,5\}$
Fault	f_{y_i}	$i \in \{1,2,3,4,5\}$

define the variable space $\Omega(G)$, where G is the complete system. The variable space is

$$\Omega(G) = \{L_i \geqslant 0, F_i \geqslant 0, u_i \geqslant 0, 0 \leqslant \chi \leqslant 1\}$$

where each L_i is a level in a tank, F_i is a flow, u_i is control signal, and χ is concentration.

11.3.3 Model

The different sub-models are described below.

Pulper and Tank Equations

The fluid in the pulper consists of two parts. The state χ is the concentration of paper mixture, i.e. when $\chi = 1$ there is only paper mixture in the tank, and when $\chi = 0$ there is only water in the tank. The pressure p is the pressure at the bottom of the tank where pipe three is connected. The pulper model is

$$\dot{L}_1 = \frac{1}{A_1}(F_1 + F_2 - F_3) \tag{11.1a}$$

$$\dot{\chi} = \frac{1}{A_1 L_1}\left(F_1(\chi_1 - \chi) + F_2(\chi_2 - \chi)\right) \tag{11.1b}$$

$$p = p_{atm} + g\rho L_1. \tag{11.1c}$$

As was mentioned in Section 11.3.1 only a part of the system is analyzed in this project. Due to this, the concentration and bottom pressure in the tank is not modeled. The tank model is therefore

$$\dot{L}_2 = \frac{1}{A_2}(F_3 - F_4). \tag{11.1d}$$

TABLE 11.2: Type definitions.

Type	Inherited type	Add. constraint	Variables
Control	Real	(min=0)	u_i
Sensor	Real		y_i
Fault	Real		f_i
Flow	SIunits.Volume-FlowRate	(min=0)	F_i, F_3^{air}, F_3^{fluid}, F_4
Height	SIunits.Height	(min=0)	L_i
Concentration	Real	(min=0, max=1)	χ, χ_i
Pressure	SIunits.Pressure	(min= 0)	p, p_{atm}, p_i
Density	SIunits.Density		ρ
Area	SIunits.Area		A_j
Acceleration	SIunits.-Acceleration		g
SIunits	Modelica.SIunits		

Pipe One and Two

The pipe equations are modeled as pressure losses due to friction δ_{fric} and control valves δ_{cv}. The pressure in the inlet to pipe one and two are here assumed to be constant. The pipe models are

$$p_{atm} - p_i = \delta_{cv_i} + \delta_{fric_i} \tag{11.2a}$$

$$\delta_{cv_i} = -b_i \frac{F_i^2}{u_i^2} \tag{11.2b}$$

$$\delta_{fric_i} = -a_i F_i^2 \tag{11.2c}$$

where $i \in \{1, 2\}$.

Pipe Three

Pipe three is equipped with a constant speed pump that give a pressure increase δ_{pump}. The model is

$$p_{atm} - p = \delta_{cv_3} + \delta_{fric_3} + \delta_{pump} \tag{11.3a}$$

$$\delta_{cv_3} = -b_3 \frac{F_3^2}{u_3^2} \tag{11.3b}$$

$$\delta_{fric_3} = -a_3 F_3^2 \tag{11.3c}$$

$$\delta_{pump} = d_1 \sqrt{1 - \left(\frac{F_3}{d_2}\right)^2}. \tag{11.3d}$$

When the level in the tank reduces to zero, there will be a flow of air going true the pipe instead of fluid. For simulation purposes, pipe 3's

flow is therefore partitioned into two parts, air F_3^{air} and fluid F_3^{fluid}. Model equations (11.3a-11.3d) are used to extract the total flow thru the pipe, while the following equations are used to partition the flow into correct parts,

$$F_3^{fluid} = \begin{cases} F_3 & \text{if } L > \epsilon \text{ or } F_3 < F_1 + F_2 \\ F_1 + F_2 & \text{otherwise} \end{cases} \tag{11.3e}$$

$$F_3^{air} + F_3^{fluid} = F_3 \tag{11.3f}$$

where $\epsilon = 2 \cdot 10^{-6}$ is chosen to be above the tolerance level of 10^{-6}.

Pipe Four

Pipe four is modeled as a constant flow. One can mention that the actual process is controlled so that the flow to the cyclone and the concentration in the pulper is constant. The flow thru pipe four is therefore relatively constant, since the flow in pipe ten is small in normal operation.

Sensor Equations

The system is equipped with five sensors. All of the sensors can be affected by unknown sensor faults and are affected by sensor noise.

$$y_1 = L_1 + f_{y_1} + v_1 \tag{11.4a}$$

$$y_2 = \chi + f_{y_2} + v_2 \tag{11.4b}$$

$$y_3 = F_3 + f_{y_3} + v_3 \tag{11.4c}$$

$$y_4 = L_2 + f_{y_4} + v_4 \tag{11.4d}$$

$$y_5 = \dot{L}_2 + f_{y_5} + v_5. \tag{11.4e}$$

The sensor-faults should be detected by the residual generators.

11.3.4 Simulation Problems with the Model

When the control signal is small, it is not possible to use the model of the pipes. The problem is that the simulation software runs into a division with zero problem when some control signal u_i tends to zero. A small positive number is therefore added to the control signals. Equation (11.2b) and (11.3b) are thereby changed to

$$\delta_{cv_i} = -b_i \frac{F_i^2}{u_i^2 + \epsilon}$$

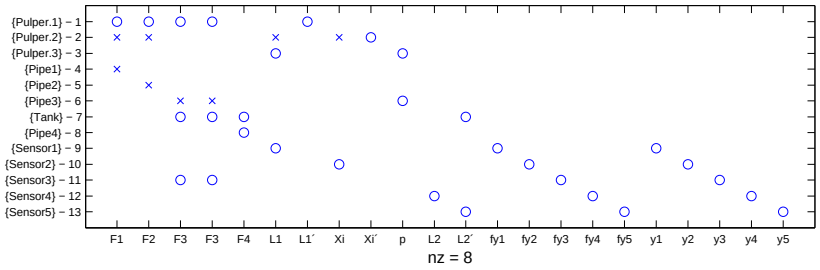


FIGURE 11.2: Structural model for the tank system.

where $\epsilon = 2 \cdot 10^{-6}$ for $i \in \{1, 2, 3\}$. This effectively limits the flows to positive values. The variable space is therefore

$$\Omega(G) = \{L_i \geqslant 0, F_i > 0, u_i \geqslant 0, 0 \leqslant \chi \leqslant 1\}. \tag{11.5}$$

11.3.5 *Noise*

The noise is band-limited white noise with mean value of zero. The standard deviation is different for the different outputs. The standard deviations are set to

$$\begin{aligned} \sigma_{y_1} &= 0.06 & \sigma_{y_2} &= 0.02 \\ \sigma_{y_3} &= 0.004 & \sigma_{y_4} &= 0.04 \\ \sigma_{y_3} &= 0.004, \end{aligned}$$

which results in a signal-to-noise ratio, SNR≈ 5. The noise is generated in MATLAB and added to the outputs.

11.4 RESIDUAL GENERATORS

Four different residual generators are implemented in this section. The complete analysis of the first two are presented, while for the last two, only the final results are presented.

11.4.1 *Sets of* MSO *Sets*

The algorithm described in Section 9.3 is given the structural model shown in Figure 11.2 as input. The notation in the figure was described in Section 10.3.

For convenience, the equations have been merged into sets of equations corresponding to the different parts of the model. Consider for

example the three equations (11.2) belonging to pipe one; these equations have been merged to one structural equation, Pipe1. Notice that this does not change the set of sets that are found, because the equations must be solved together. In for example Pipe1, the set of equations (11.2) must be solved together since the two variables δ_{cv_1} and δ_{cv_2} are included in all three equations, while not being included in any other equation. Therefore, the two last equations must be used to solve for these two variables, while the first equation cannot be included in an MSO set, without the two other also being included. Only the equations that structurally must be used together have been merged.

The MSO sets that were found are shown in Figure 11.3. In the figure, a circle $\circ$ in row j and column i means that the equations in row i includes the fault in column j, i.e. ideally a residual generator using this equation will be sensitive to these faults. The four MSO sets that are chosen to be implemented are number 2, 6, 12, and 13 in the figure. These are chosen such that all faults are detectable and that fault f_{y_1} is isolatable, while the rest of the faults are isolatable from all faults with exception to f_{y_1}.

For example, the sixth MSO set in the figure use the relationship between the flow in pipe one and two, the concentration in the pulper (Pulper.2), and sensor one and two. The MSO set includes the dynamic state χ and will be denoted MSO_2.

The influence matrix for the chosen MSO sets is given in Table 11.3. Notice that all faults are detectable and that fault f_{y_1} is completely isolatable. Faults f_{y_2}, f_{y_3}, f_{y_4}, and f_{y_5} are isolatable from each other but not from fault f_{y_1}.

TABLE 11.3: Influence matrix.

MSO set	f_{y_1}	f_{y_2}	f_{y_3}	f_{y_4}	f_{y_5}	Equation set
$MSO_1(2)$	X	0	X	0	0	{S.1, S.3, Pipe3, Pulper.3}
$MSO_2(6)$	X	X	0	0	0	{S.1, Pipe1, Pipe2, S.2, Pulper.2}
$MSO_3(12)$	X	0	0	X	0	{S.1, S.4, Pulper.3, Pipe3, Tank, Pipe4}
$MSO_4(13)$	X	0	0	0	X	{S.1, S.5, Pulper.3, Pipe3, Tank, Pipe4}

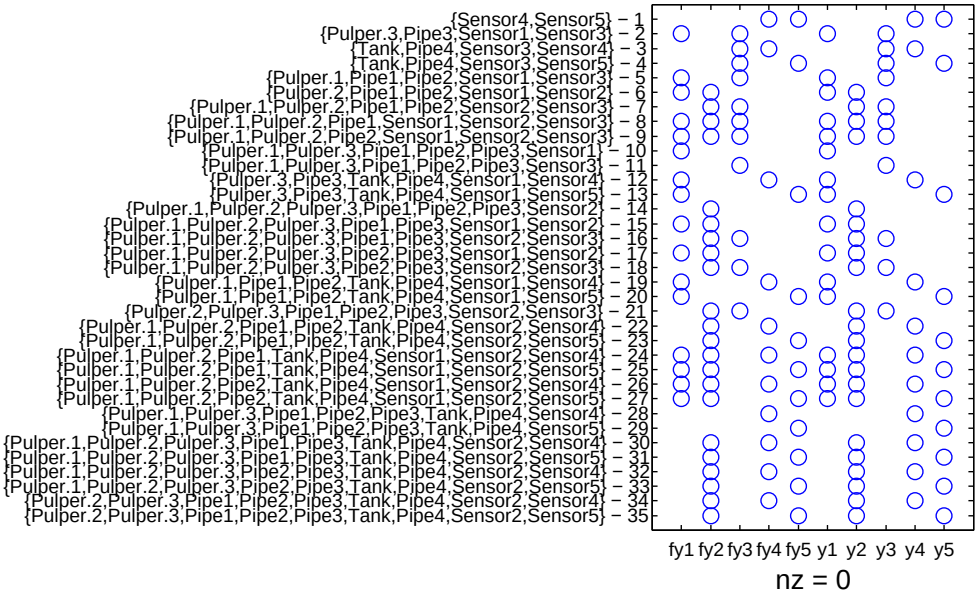


FIGURE 11.3: MSO sets found from the structural model.

11.4.2 Residual Generator One

The MSO model for the first MSO set is

$$g = \begin{bmatrix} p - p_{atm} - g\rho L_1 \\ p_{atm} - p - \delta_{cv_3} - \delta_{fric_3} - \delta_{pump} \\ \delta_{cv_3} + b_3 \frac{F_3^2}{u_3^2} \\ \delta_{fric_3} + a_3 F_3^2 \\ \delta_{pump} - d_1 \sqrt{1 - \left(\frac{F_3}{d_2}\right)^2} \\ y_1 - L_1 \\ y_3 - F_3 \end{bmatrix} + \Gamma R = 0.$$

All equations are structurally redundant. To check if they are analytically redundant, the Jacobian is calculated

$$J = \frac{\partial g}{\partial [p, F_3, L_1, \delta_{cv_3}, \delta_{fric_3}, \delta_{pump}, r^{(n)}]^T}$$
$$= \left[\begin{array}{cccccc|c} 1 & 0 & -(g\,\rho) & 0 & 0 & 0 & \\ -1 & 0 & 0 & -1 & -1 & -1 & \\ 0 & \frac{2\,b_3\,F_3}{{u_3}^2} & 0 & 1 & 0 & 0 & \\ 0 & 2\,a_3\,F_3 & 0 & 0 & 1 & 0 & \\ 0 & \frac{d_1\,F_3}{{d_2}^2\sqrt{1-\frac{{F_3}^2}{{d_2}^2}}} & 0 & 0 & 0 & 1 & \bar{\Gamma} \\ 0 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & 0 & \end{array}\right]$$

and the Jacobian determinant is

$$|J| = \xi(\cdot)\bar{\Gamma} = \left[1, 1, 1, 1, 1, -g\rho, \left(2a_3F_3 + \frac{d_1F_3}{d_2^2\sqrt{1-\frac{F_3^2}{d_2^2}}} + \frac{2b_3F_3}{u_3^2}\right)\right]\bar{\Gamma}.$$

Since the variable space was limited to positive flows (11.5), all equations are analytically redundant.

The system is static and is therefore stable. A design choice is to use the first equation to define the residual variable. With $\Gamma_{11} = 10^5$, $\Gamma_{12} = \alpha\Gamma_{11}$, and $\alpha = 2$ a stable residual generator is formed.

Simulations showed that the equations are sensitive to noise when L_1 is small. Therefore an if-case is added to the residual generator. The if-case is similar to the one used for pipe 3 in (11.3e). The implementation is

```
if (y1 >= beta_y1 and u3 >= beta_u3) then;
  p = patm + g*rho*L1 + 1e5*der(res) + 1e5*alpha*res;
else;
  0 = 1e5*der(res) + 1e5*alpha*res;
end if;
```

i.e. the residual tends to zero when the level in the tank or the flow out of the tank is to low according to sensor y_1 and actuator signal u_3. In the simulations the constants $\beta_{y_1} = 0.2$ and $\beta_{u3} = 0.005$ are used.

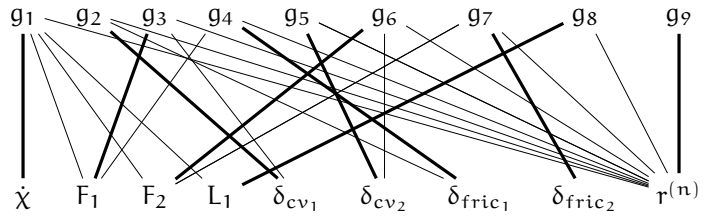


FIGURE 11.4: Perfect matching for the second MSO model.

11.4.3 Residual Generator Two

The model formed from the second MSO set is

$$g = \begin{bmatrix} \dot{\chi} - \frac{1}{AL_1}\left(F_1(\chi_1 - \chi) + F_2(\chi_2 - \chi)\right) \\ p_{atm} - p_1 - \delta_{cv_1} - \delta_{fric_1} \\ \delta_{cv_1} + b_1 \frac{F_1^2}{u_1^2} \\ \delta_{fric_1} + a_1 F_1^2 \\ p_{atm} - p_2 - \delta_{cv_2} - \delta_{fric_2} \\ \delta_{cv_2} + b_2 \frac{F_1^2}{u_2^2} \\ \delta_{fric_2} + a_2 F_2^2 \\ y_1 - L_1 \\ y_2 - \chi \end{bmatrix} + \Gamma R = 0.$$

A perfect matching is shown in Figure 11.4. It can be seen that the last equation is structurally redundant and that no other equations are structurally redundant. To see if it is analytically redundant, calculate the Jacobian

$$J = \frac{\partial g}{\partial[\dot{\chi}, F_1, F_2, L_1, \delta_{cv_1}, \delta_{cv_2}, \delta_{fric_1}, \delta_{fric_2}, r^{(n)}]^T} =$$

$$\left[\begin{array}{cccccccc|c} 1 & -\frac{-\chi+\chi_1}{A_1 L_1} & -\frac{-\chi+\chi_2}{A_1 L_1} & \frac{F_1(\chi_1-\chi)+F_2(\chi_2-\chi)}{A_1 L_1^2} & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & \\ 0 & \frac{2 b_1 F_1}{{u_1}^2} & 0 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 2 a_1 F_1 & 0 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & \bar{\Gamma} \\ 0 & 0 & \frac{2 b_2 F_2}{{u_2}^2} & 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 2 a_2 F_2 & 0 & 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \end{array}\right]$$

and the Jacobian determinant is

$$|J| = \left[0, \ldots, 0, \left(-4 a_1 a_2 F_1 F_2 - \frac{4 a_2 b_1 F_1 F_2}{u_1^2} - \frac{4 a_1 b_2 F_1 F_2}{{u_2}^2} - \frac{4 b_1 b_2 F_1 F_2}{{u_1}^2 {u_2}^2}\right) \bar{\Gamma}_9\right].$$

Also here, the equation is analytically redundant since the variable space was limited to positive flows.

A design choice is to let $\Gamma_{11} = K\Gamma_{91}$ and $\Gamma_{12} = K\Gamma_{92}$, where K is some constant. Transformation of the dynamic and the analytically redundant equation to frequency domain give

$$\chi = \frac{\frac{1}{A_1 L_1}(F_1\chi_1 + F_2\chi_2)}{s + \frac{F_1+F_2}{A_1 L_1} + K}.$$

The residual generator is stable if $K > 0$. The parameters $\Gamma_{91} = 1$, $\Gamma_{92} = 2$, and $K = 0.1$ have been used in the simulations.

Similar to the first residual generator, this generator is also sensitive to noise when L_1 is small. The residual variable is therefore defined by

```
if L1 >= beta then;
  y2 = Xi + der(res) + alpha*res;
else;
  0 = der(res) + alpha*res;
end if;
```

where the residual tends to zero when the level in the tank is greater or equal to $\beta = 0.2$.

11.4.4 Residual Generator Three

Residual generator three uses an estimation of the flow thru pipe three and relates this to the height in the tank. The dynamic state is L_2 and sensor equation (11.4d) is analytically redundant.

11.4.5 Residual Generator Four

Residual generator four is similar to generator three, with the difference that it uses sensor equation (11.4e). The MSO model is static and all equations are analytically redundant.

11.4.6 Thresholds

If a residual's value exceeds some stated threshold then an alarm is generated. The threshold should be sufficiently high to reduce the probability for false alarms, while being sufficiently low so that the probability for missed detections are low.

The thresholds are here set such that the probability for false alarm is 0.02 %. The noise is considered to be band limited white noise when the thresholds are determined. A fault-free simulation is performed

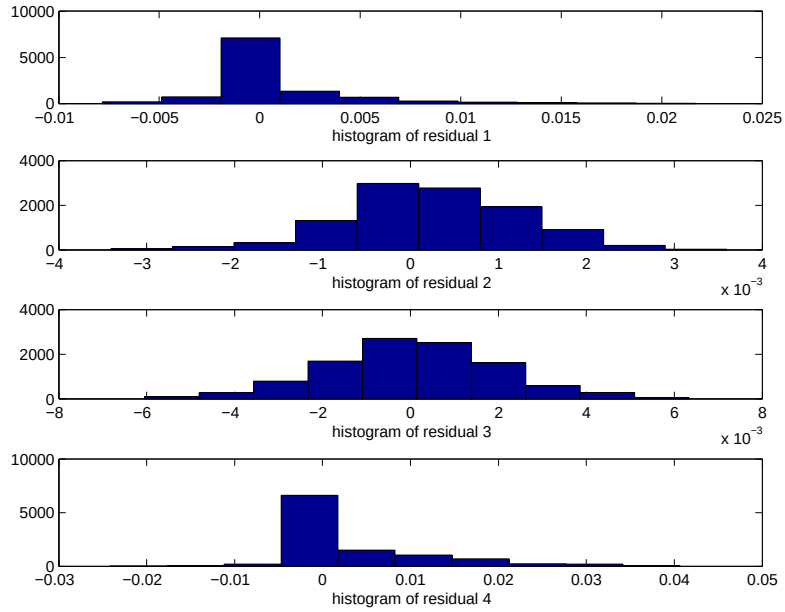


FIGURE 11.5: Histogram of the residuals from fault-free simulations.

and used to calculate the thresholds. Figure 11.5 shows histograms of the four residuals in the fault-free case. It can be seen that the noises are approximately normal distributed for all residuals.

The probability is determined from the cumulative normal distribution $F(x)$ for the normal distribution $N(0, \sigma)$. The probability $P(Alarm|NF) = 2{\cdot}10^{-4}$, where NF means no-fault, therefore the threshold T should be chosen such that

$$F(T) = 1 - \frac{2 \cdot 10^{-4}}{2}.$$

The standard deviations for the noises are determined from the fault-free simulations. The thresholds are used to normalize the residuals such that the thresholds are set to ± 1.

11.5 SIMULATIONS

In the following pages, plots from different simulations are shown. The figures show, sensor-values, control-signals, faults, and residual values. Included are also the logic implications of the residuals' values, i.e. the diagnoses.

High Excitation Simulation

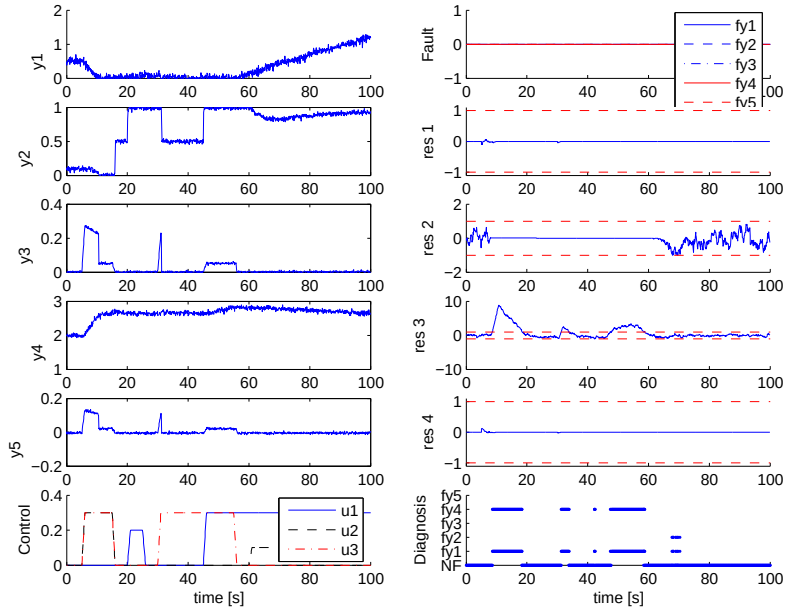


FIGURE 11.6: High excitation of the states and no faults.

A simulation with high excitation of the state variables is performed to test the model. The result is shown in Figure 11.6. Notice that the tank is emptied around time 10 s. In the plot of sensor y_2 it can be seen how the concentration is changed between 0 and 1. The control signals are shown in the lowest left plot and the fault signals in the upper right plot.

Since the level in the tank is low in the time interval 10 to 60 s, residual one, two, and four tends to zero during this time. As can be seen in the plot of residual three, the results from this residual generator is faulty during this test. Therefore, it is necessary to either change the residual's definition, or to use an adaptive threshold that compensates for the bad behavior of the residual. This has however not been done in these simulations. All residual generators except the third are below the threshold during the simulation.

The diagnoses are shown in the lowest right plot. At for example time 0 s is the diagnosis NF (no fault), and at time 15 s it the diagnoses f_{y_4} or f_{y_1}.

Fault-Free Simulation

The results from a fault-free simulation are shown in Figure 11.7. The rest of the simulations will use the same control signals as is used in this simulation. As should be, the residuals' values are mostly below the thresholds during the simulation. There are some minor false alarms, notably in the later part of the simulation. The false alarms should be approximately 0.02 % of the samples, see Section 11.4.6.

Fault f_{y_1}

The result from a simulation where f_{y_1} has been added is shown in Figure 11.8. In the Figure, it can be seen that residual one and four do not react until F_3^{Fluid} deviates significantly from zero; compare sensor y_3 with the residuals at time 60 s.

Not all residuals react strongly enough, with the result that f_{y_1} can not be completely isolated. This can for example be seen in the plot of residual three that is lowered when the fault is induced, but it is not lowered such that it goes below the threshold. Residual two can be seen to only weakly detect the fault.

Fault f_{y_2}

Fault f_{y_2} has been induced in the simulation shown in Figure 11.9. Residual two should react which is also the case. One can notice that it reacts weakly on the induced fault. Since fault f_{y_2} can not be isolated from fault f_{y_1}, the diagnosis is that f_{y_1} or f_{y_2} are faulty. This can be seen in the right lowest plot.

Fault f_{y_3}

Fault f_{y_3} has been induced in the simulation shown in Figure 11.10. As it should be, residual one reacts to the fault. Notice that the residual reacts strongly to the fault.

Fault f_{y_4}

Fault f_{y_4} has been induced in the simulation shown in Figure 11.11. As it should be, residual three reacts to the fault. The residual reacts weakly on the induced fault.

Fault f_{y_5}

Fault f_{y_5} has been induced in the simulation shown in Figure 11.12. As it should be, residual four reacts to the fault. The residual reacts strongly to the fault.

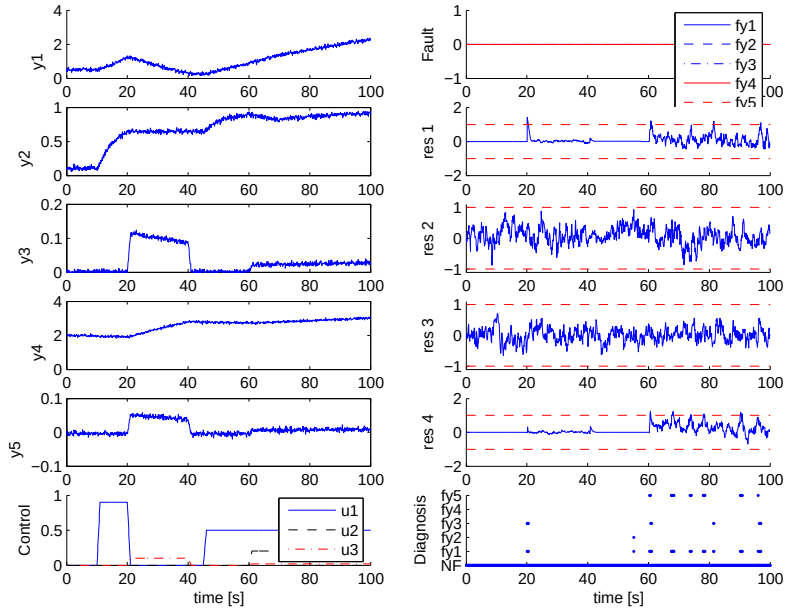


FIGURE 11.7: Result when no fault has been induced in the system.

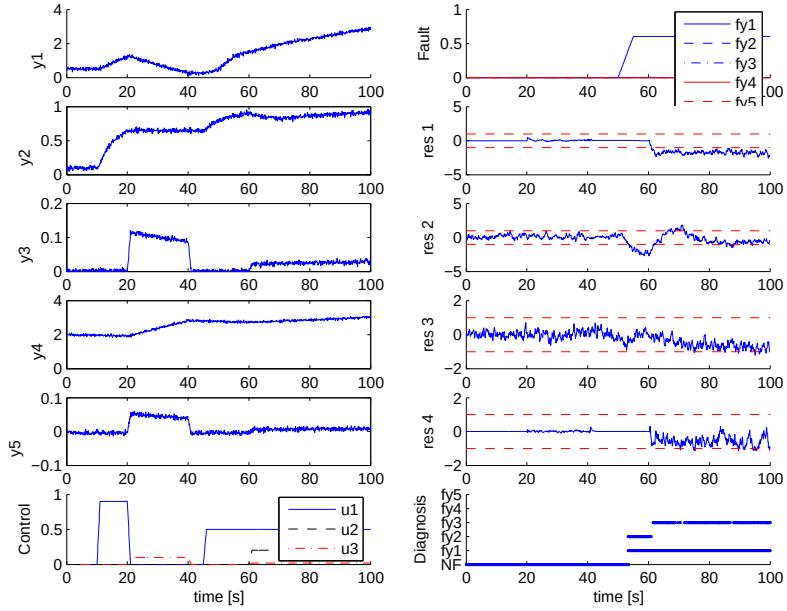


FIGURE 11.8: Result when fault f_{y_1} has been induced.

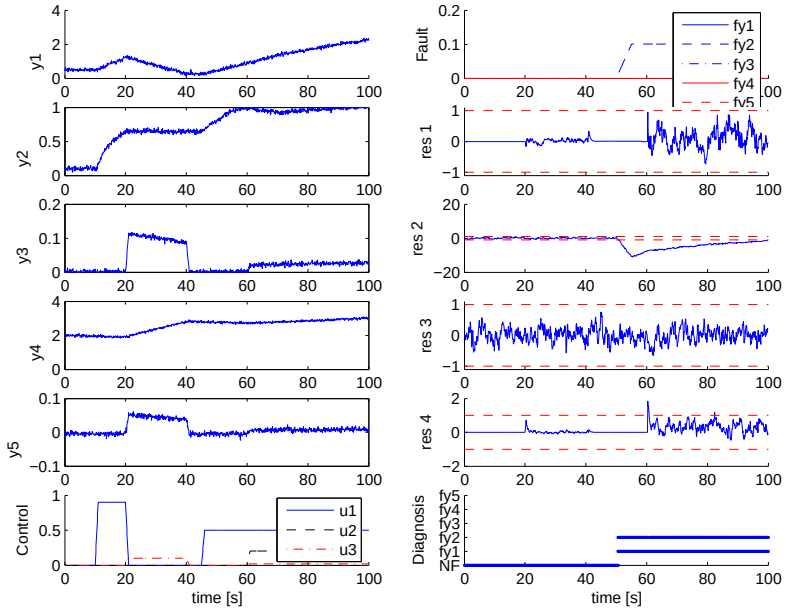


FIGURE 11.9: Result when fault f_{y_2} has been induced.

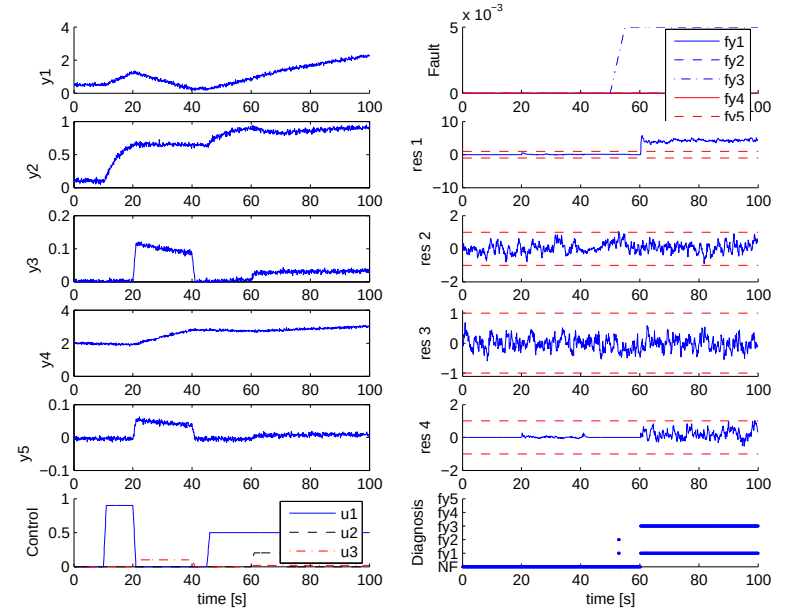


FIGURE 11.10: Result when fault f_{y_3} has been induced.

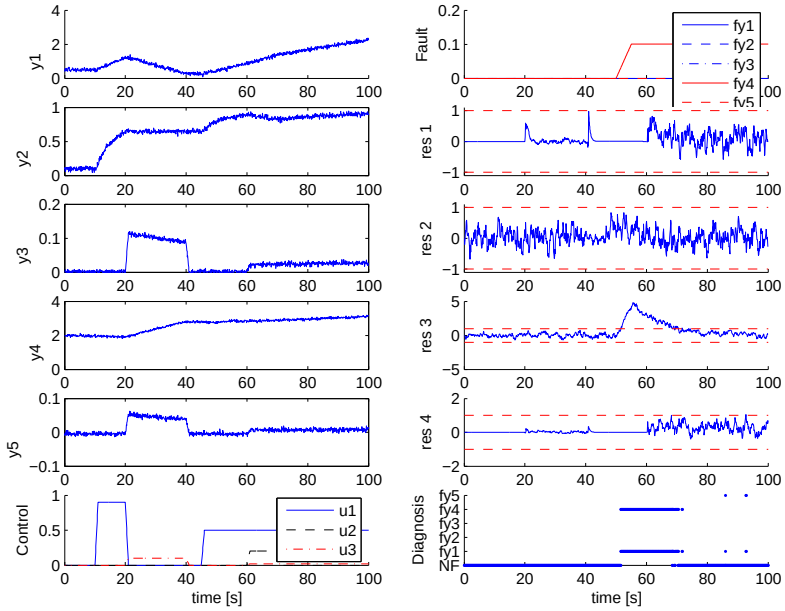


FIGURE 11.11: Result when fault f_{y_4} has been induced.

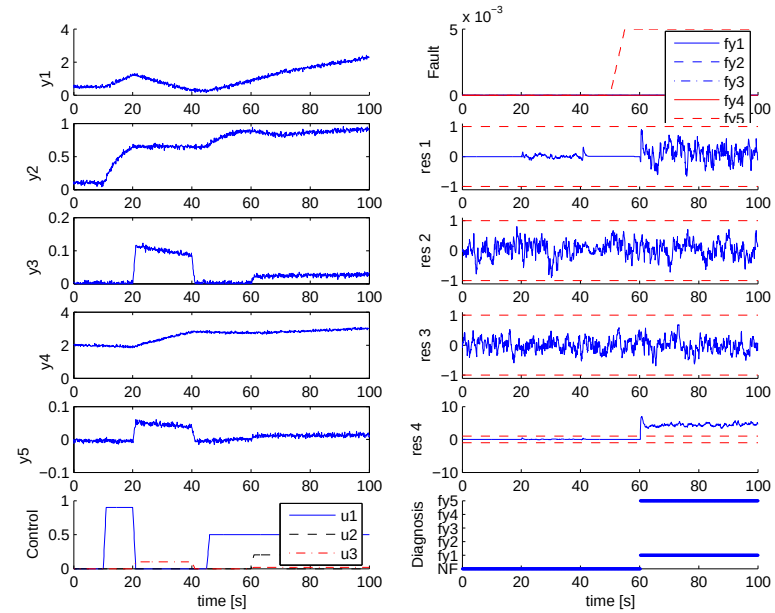


FIGURE 11.12: Result when fault f_{y_5} has been induced.

12

Conclusions Part II

Residual generators have been designed from minimally structurally overdetermined sets of equations with an extra residual variable, denoted MSO models. Instead of analytically transforming the MSO models from DAEs into some specific residual generator form, simulation was used. The result is simulation based residual generators. Since it is not possible to add the residual variable arbitrary, constraints on how the residual variable could be added have been stated. The constraints can be partitioned into two parts. Firstly, the residual variable must be added to an analytically redundant equation. Secondly, it must be added in such a way that the MSO model is stable.

The relationship between approximations of derivatives of sensor values and evaluation complexity has been studied. It has been shown that it is possible to obtain simplifications of the MSO models, at the expense of approximations of derivatives of sensor values. If approximations of derivatives are allowed, then high index MSO models can often be relaxed to lower index MSO models.

In Chapter 10, the method was demonstrated on a satellite system. It was exemplified how different residual generators could be designed depending on which derivatives of sensor values that were approximated. It was in Chapter 11 demonstrated how the method can take advantage of a DAE simulator like Dymola, which uses the modeling language Modelica. In this non-trivial example from a paper mill, it was demonstrated how to extract the equations from the implemented model, how to add a residual variable, and how to simulate the resulting residual generator.

REFERENCES

[ALW+03] M. Albert, T. Längle, H. Wörn, M. Capobianco, and A. Brighenti, *Multi-agent systems for industrial diagnostics*, Proceedings of IFAC Safeprocess'03 (Washington, USA), 2003.

[AP98] U. M Ascher and L. R Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*, siam, 1998.

[ATL02] Martin Albert and Heinz Wörn Thomas Längle, *Development tool for distributed monitoring and diagnosis systems*, 13th Workshop on Principles of Diagnosis, DX'02 (Semmering, Austria), 2002.

[Axe04] Tobias Axelsson, *Diagnosis system conceptual design utilizing structural methods - applied on a UAV's fuel system*, Master's thesis, Linköpings Universitet, SE-581 83 Linköping, 2004.

[BCF+04] Jonas Biteus, Gunnar Cedersund, Erik Frisk, Mattias Krysander, and Lars Nielsen, *Improving airplane safety by incorporating diagnosis into existing safety practice*, Tech. Report LiTH-R-2648, Dept. of Electrical Engineering, Linköpings universitet, Sweden, 2004.

[Bit01] Jonas Biteus, *Diagnosis of fluid systems utilizing gröbner bases and filtering of consistency relations*, Master's thesis, Linköpings Universitet, SE-581 83 Linköping, 2001.

[Bit02] _____, *Mean value engine model of a heavy duty diesel engine*, Tech. Report LiTH-ISY-R-2666, Dept. of Electrical Engineering, Linköpings universitet, Sweden, 2002.

[BJN04] Jonas Biteus, Mathias Jensen, and Mattias Nyberg, *Decentralized diagnosis in heavy duty vehicles*, CCSSE (Norrköping, Sweden), 2004.

[BJN05] _____, *Distributed diagnosis for embedded systems in automotive vehicles*, IFAC World Congress (Praha, Czech Republic), 2005.

[BN02] Jonas Biteus and Mattias Nyberg, *Dynamic evaluation of minimal structurally singular sets*, CCSSE (Norrköping, Sweden), 2002.

[BN03] _____, *Residual generators for DAE systems utilizing minimal subsets of model equations*, 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS (Washington, D.C., U.S.A.), June 2003.

[Bre96] Gerhard Brewka (ed.), *Reasoning in description logics*, Studies in Logic, Language and Information, CSLI Publications, 1996.

[CG95] Stephen L. Campell and C. William Gear, *The index of general nonlinear DAEs*, Numerische Mathematik (1995), no. 72, 173–196.

[CP99] Jie Chen and R.J. Patton, *Robust model-based fault diagnosis for dynamic systems*, Kluwer Academic Publishers, 1999.

[dK92] John de Kleer, *Focusing on probable diagnoses*, Readings in Model-Based Diagnosis (W. Hamscher, L. Console, and John de Kleer, eds.), Morgan Kaufmann Publishers, 1992.

[dKMR92] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter, *Characterizing diagnoses and systems*, Artificial Intelligence **56** (1992), no. 2-3.

[Elm02] Hilding Elmqvist, *Dymola, user's manual*, Dynasim AB, Lund, Sweden, 4.2b ed., 2002.

[FÅ05] Erik Frisk and Jan Åslund, *An observer for semi-explicit differential-algebraic systems*, Proceedings of IFAC World Congress (Prague, Czech Republic), July 2005.

[FDKC03] Erik Frisk, Dilek Düştegör, Mattias Krysander, and Vincent Cocquempot, *Improving fault isolability properties by structural analysis of faulty behavior models: application to the DAMADICS benchmark problem*, Proceedings of IFAC Safeprocess'03 (Washington, USA), 2003.

[Fri01] Erik Frisk, *Residual generation for fault diagnosis*, Ph.D. thesis, Linköpings Universitet, November 2001.

[Fri04] Peter Fritzson, *Principles of object oriented modeling and simulation with modelica 2.1*, Wiley, 2004.

[Har69] Frank Harary, *Graph theory*, Addison-Wesley Publishing Co., Boston, U.S.A., 1969.

[Hay99] C.C. Hayes, *Agents in a nutshell-a very brief introduction*, Knowledge and Data Engineering, IEEE Transactions on **11** (1999), no. 1, 127–132.

[HCK92] W. Hamscher, L. Console, and John Kleer, de (eds.), *Readings in model-based diagnosis*, Morgan Kaufmann Publishers, 1992.

[HVL05] D. Hristu-Varsakelis and W. S. Levine, *The handbook of networked and embedded control systems*, Springer Verlag, 2005, Contributions by Lars Nielsen.

[KKZ02] J. Kurien, X. Koutsoukos, and F. Zhao, *Distributed diagnosis of networked, embedded systems*, Thirteenth International Workshop on Principles of Diagnosis (Semmering, Austria), May 2002.

[KN02] Mattias Krysander and Mattias Nyberg, *Structural analysis utilizing MSS sets with application to a paper plant*, 13th International Workshop on Principles of Diagnosis (Semmering, Austria), 2002.

[Kry03] Mattias Krysander, *Design and analysis of diagnostic systems utilizing structural methods*, Tech. report, Vehicular Systems, Dept. of Electrical Engineering, 2003, LiU-TEK-LIC-2003:37, Thesis No. 1038.

[KS03] et al. Köppen-Seliger, *Magic: An integrated approach for diagnostic data management and operator support*, Proceedings of IFAC Safeprocess'03 (Washington, USA), 2003.

[KÅN05] Mattias Krysander, Jan Åslund, and Mattias Nyberg, *An efficient algorithm for finding over-constrained sub-systems for construction of diagnostic tests*, 16th Workshop on Principles of Diagnosis, DX'05 (California, USA), June 2005.

[LS01] J. Lunze and J. Schröder, *State observation and diagnosis of discrete-event systems described by stochastic automata*, Discrete Event Dynamic Systems **11** (2001), no. 4, 319–369.

[Mat05] The Mathworks Inc., Natick, Massachusetts, U.S.A., *Matlab*, 7.0 ed., 2005.

[MS92] Sven Erik Mattsson and Gustaf Söderlind, *A new technique for solving high-index DAE using dummy derivatives*, Computer-Aided Control System Design (Napa, USA), IEEE, March 1992, pp. 218–224.

[NF05] Mattias Nyberg and Erik Frisk, *Model based diagnosis of tehcnical processes*, Printed by the Institution of Electrical Engineering, Linköpings universitet, 2005.

[Nyb99] Mattias Nyberg, *Model based fault diagnosis: Methods, theory, and automotive engine applications*, Ph.D. thesis, Linköpings Universitet, June 1999.

[Par98] Van Dyke Parunak, H., *What can agents do in industry, and why? an overview of industrially-oriented R&D at CEC*, Lecture Notes in Computer Science **1435** (1998).

[PFC00] Ron Patton, Paul Frank, and Robert Clark (eds.), *Issues of fault diagnosis for dynamic systems*, Springer, 2000.

[PI01] Claudio De Persis and Alberto Isidori, *A geometric approach to nonlinear fault detection and isolation*, IEEE Trans. on Automatic Control **46** (2001), no. 6, 853–865.

[Pro02] Gregory Provan, *A model-based diagnosis framework for distributed systems*, Proc. of the Thirteenth International Workshop on Principles of Diagnosis (Semmering, Austria), May 2002.

[Red03] Red hat, Inc., Raleigh, North Carolina, U.S.A., *Red hat*, 9 ed., 2003.

[Rei87] Raymond Reiter, *A theory of diagnosis from first principles*, Artificial Intelligence **32** (1987), no. 1, 57–95.

[RTF03] X. Ren, H. A. Thompson, and P. J. Fleming, *Intelligent agents for distributed fault diagnosis*, Proceedings of IFAC Safeprocess'03 (Washington, USA), 2003.

[RTW03] Nico Roos, ten Teije, Annette, and Cees Witteveen, *A protocol for multi-agent diagnosis with spatially distributed knowledge*, 2nd Conference on Autonomous Agents and Mult-Agent Systems (Australia), July 2003.

[Rug96] Wilson J. Rugh, *Linear system theory*, 2nd ed., Prentice Hall, Upper Saddle River, USA, 1996.

[V+95] Vaithianathan Venkatasubramanian et al., *Local bifurcations and feasibility regions in differential-algebraic systems*, IEEE Transactions on Automatic Control **AC-40** (1995), no. 12, 1992–2013.

[Wot01] Franz Wotawa, *A variant of reiter's hitting-set algorithm*, Information Processing Letters (2001), no. 79, 45–51.

[ÅBF+05] Jan Åslund, Jonas Biteus, Erik Frisk, Mattias Krysander, and Lars Nielsen, *A systematic inclusion of diagnosis performance in fault tree analysis*, IFAC World Congress (Praha, Czech Republic), 2005.

Notation

Acronyms

Type	*Symbol*	*Description*
Acronyms	ECU	Electronic control unit
	DTC	Diagnostic trouble code
	CAN	Controller area network
	MCMD	Minimal cardinality module diagnoses
	FDI	Fault detection and isolation
	ODE	Ordinary differential equations
	DAE	Differential algebraic equations

Symbols

Type	*Symbol*	*Description*
Symbols	$\mathcal{O}$	Ordo
	$\subsetneq$	Proper subset
	$\neg$	Not
	$\bot$	False
	$\top$	True
	$\{\}$	Unordered set
	$()$	Ordered set

Type	*Symbol*	*Description*
	$\exists$	Exists
	$\wedge$	Conjunction (and)
	$\vee$	Disjunction (or)
	$\lvert\cdot\rvert$	Cardinality (size)

Part I

Type	*Symbol*	*Description*
Acronyms	$\mathtt{mc}$	Minimal cardinality
	$\mathtt{ec}$	Extended cardinality
	$\mathtt{mec}$	Minimal extended cardinality
	$\mathtt{mod}$	Module
	MCMD	minimal cardinality module diagnoses
Sets	$\mathcal{X}$	Sets, often calligraphic letters
	$\mathtt{x}$	Elements, often lower case letters
Model	$\mathcal{A}$	Agents
	A	Agent
	$\mathcal{C}$	Components
	$\mathtt{c}$	Component or object
	IN	Inputs
	$\mathcal{I}$	Sub-set of inputs
	$\mathtt{i}$	Input
	OUT	Outputs
	$\mathcal{P}$	Sub-set of outputs
	σ	Output
	Θ	Objects
	θ	Object
	$\mathsf{A}, \mathsf{B}, \mathsf{C}, \ldots$	Components in examples
System	STRU	Structural descriptions
	SD	System description
	OBS	Observations
	SE	Assumptions and equations
	E	Equations
	CR	Computation requirements
	TEST	Tests
	TC	Test conditions

Type	*Symbol*	*Description*
Diagnoses	$\mathcal{D}$	Global diagnoses
	$\mathbb{D}$	Local diagnoses
	D	Diagnosis
	Π	Conflicts
	π	Conflict
	$\mathcal{S}$	Hitting sets
	S	Hitting set
Modes	$\mathtt{mode}(M, c)$	Object c is in mode M
	$AB(c)$	Abnormal, $\mathtt{mode}(AB, c)$
	$A(c)$	$AB(c)$ or $\neg AB(c)$
Functions	$\min_S(\cdot)$	Minimal set representation
	$\mathrm{con}(\cdot)$	Connections
	$\mathrm{con}_A(\cdot)$	Connection from agents
	$\mathrm{ass}(\cdot)$	Assumptions
	$\mathrm{dep}(\cdot)$	Dependency
	$\varphi_{\mathcal{C}}(\cdot)$	Complete component representation
	$\simeq$	Equal considering minimal diagnoses
	$\overset{\varphi_{\mathcal{C}}}{=}$	$\varphi_{\mathcal{C}}(\cdot)$ equality
	$\bowtie$	Merge
	rx	Received transmission
	tx	Transmitted transmission
	Ω^A	Virtual component from agent A
	S^A	Supporting information from agent A
	$\Psi_\Omega(\cdot)$	Replacement of all Ω
	$\Psi_S(\cdot)$	Replacement of all S
MCMDs	$\mathrm{informed}$	Informed variables
	$\mathcal{L}$	Lower limit
	$\mathcal{U}$	Upper limit

Part II

Type	*Symbol*	*Description*
Acronyms	MSO	Minimal structurally overdetermined
Model	G	Th model consisting of of DAEs
	g	Sub-set G
	$\mathcal{X}$	Dynamic state variables
	$\mathcal{Z}$	Instantaneous state variables
	$\mathcal{U}$	Known variables

Type	*Symbol*	*Description*
	$\mathcal{Y}$	Sensor variables
	$\mathcal{F}$	Fault variables
	x, z, u, y, f	Subsets of $\mathcal{X}, \mathcal{Z}, \mathcal{U}, \mathcal{Y}, \mathcal{F}$ respectively
Other variables	J	Jacobian
	Ω	Variable space
	ξ	Vector in the Jacobian determinant
	Γ	Parameter matrix for R
	r	Residual variable
	R	$[r, \dot{r}, \ldots, r^{(n)}]$
Functions	$\mathtt{ass}(\cdot)$	Assumptions