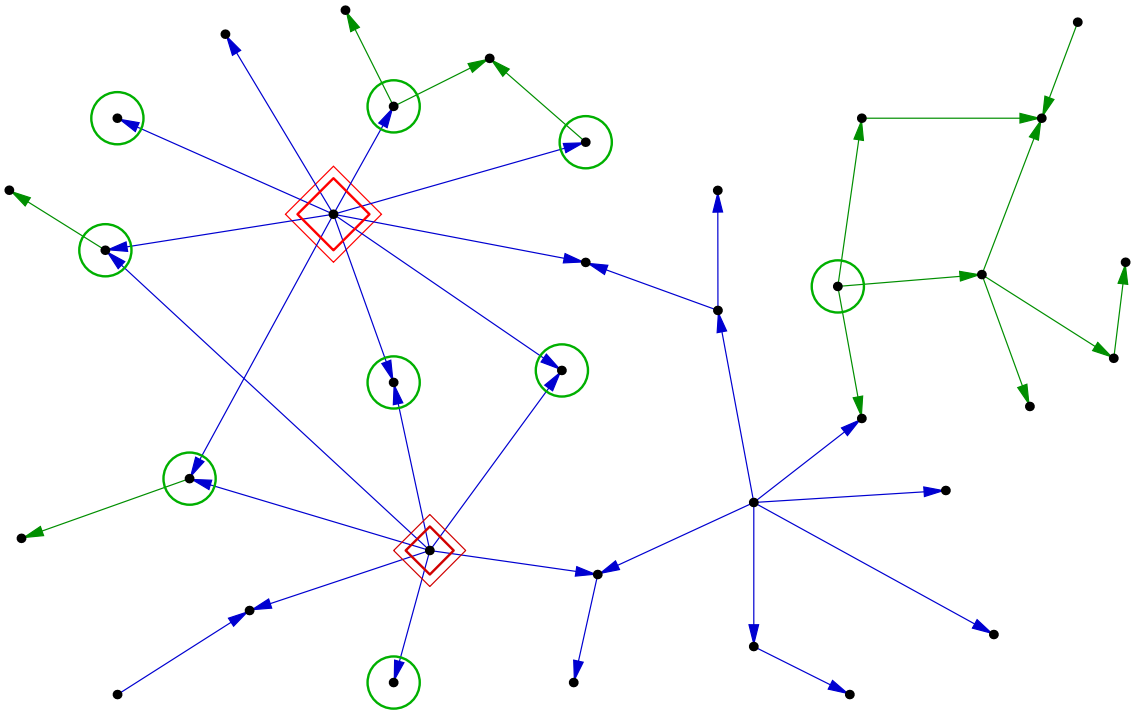
Linköping Studies in Science and Technology
Dissertation No. 1074

# Fault Isolation in Distributed Embedded Systems

Jonas Biteus



Vehicular Systems
Department of Electrical Engineering
Linköpings universitet, Linköping, Sweden

Linköping 2007

# FAULT ISOLATION IN DISTRIBUTED EMBEDDED SYSTEMS

Jonas Biteus

Linköpings universitet
Sweden

**Cover page:** The cover page illustrates a diagnostic system in a heavy duty vehicle. Each dot represents a diagnostic test that supervises some components. An arrow is drawn from a test if it supervises all components supervised by the test that the arrow points at. The tests that are circled have responded and all tests that the green (gray) arrows point at can therefore be removed. Due to the responded tests, some components are suspected to be faulty and to be certain that they are faulty, tests that are squared should be evaluated. Further, among these squared tests, it is best to first evaluate the highlighted test.

FAULT ISOLATION IN DISTRIBUTED EMBEDDED SYSTEMS



E-mail: biteus@isy.liu.se
Homepage: www.vehicular.isy.liu.se

Vehicular Systems
Department of Electrical Engineering
Linköpings universitet
SE – 581 83 Linköping
Sweden

ISBN 978-91-85715-66-4
ISSN 0345-7524

# ABSTRACT

To improve safety, reliability, and efficiency of automotive vehicles and other technical applications, embedded systems commonly use fault diagnosis consisting of fault detection and isolation. Since many systems are constructed as distributed embedded systems including multiple control units, it is necessary to perform global fault isolation using for example a central unit. However, the drawbacks with such a centralized method are the need of a powerful diagnostic unit and the sensitivity against disconnections of this unit.

Two alternative methods to centralized fault isolation are presented in this thesis. The first method performs global fault isolation by a distributed sequential computation. For a set of studied systems, the method gives, compared to a centralized method, a mean reduction in maximum processor load on any unit with 40 and 70 % for systems consisting of four and eight units respectively. The second method instead extends the result of the local fault isolation performed in each unit such that the results are globally correct. By only considering the components affecting each specific unit, the extended result in each agent is kept small. For a studied automotive vehicle, the second method gives, compared to a centralized method, a mean reduction in the sizes of the results and the maximum processor load on any unit with 85 and 90 % respectively.

To perform fault diagnosis, diagnostic tests are commonly used. If the additional evaluation of tests can not improve the fault isolation of a component then the component is ready. Since the evaluation of a test comes with a cost in for example computational resources, it is valuable to minimize the number of tests that have to be evaluated before readiness is achieved for all components. A strategy is presented that decides in which order to evaluate tests such that readiness is achieved with as few evaluations of tests as possible.

Besides knowing how fault diagnosis is performed, it is also interesting to assess the effect that fault diagnosis has on for example safety. Since fault tree analysis often is used to evaluate safety, this thesis contributes with a systematic method that includes the effect of fault diagnosis in fault trees. The safety enhancement due to the use of fault diagnosis can thereby be analyzed and quantified.

**Keywords:** Fault diagnosis; Fault isolation; Distributed diagnosis; Embedded systems; Fault tree analysis.

# 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, supervisor, and coauthor *Lars Nielsen*, for letting me perform this work at the division. I would also like to thank my supervisor and coauthor *Erik Frisk* for many fruitful discussions about the topics in the thesis, and for spending many hours proofreading the thesis.

Thanks goes to my coauthor *Mattias Nyberg* who lead me into the research area of diagnosis, and whom I have performed collaborative work with. To my coauthor *Jan Åslund* for collaborative work and for proofreading the thesis. To my coauthor *Mattias Krysander* for collaborative work and for discussions about diagnosis.

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

To *Scania* for the automotive application. To *Mathias Jensen* and *Dan Hallgren*, for discussions about distributed diagnosis. To *Magnus Adolfson*, *David Elfvik*, and *Anna Pernestål* for general discussions about diagnosis in automotive vehicles.

This work has been supported by The Swedish Foundation for Strategic Research (Stiftelsen för Strategisk Forskning) and NFFP (The Swedish Aviation Engineering Research Programme). The Swedish Foundation for Strategic Research has supported the work through 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*
In a snow covered Linköping
February 2007

# CONTENTS

# Chapter 1

# Introduction

There are an increasing number of applications that use embedded software for control. To improve safety, reliability, and efficiency of such embedded systems, there is an increasing demand for fault diagnosis, i.e. to detect and isolate abnormally behaving components, see for example (Isermann, 2005) where automotive fault diagnosis is discussed. Fault diagnosis is performed by dedicated diagnostic systems, and the results are typically used to make autonomous decisions such as fault tolerant control (FTC), to inform the user, or for repair and maintenance.

The dominant methodology for fault diagnosis in the AI field is so called consistency based diagnosis (Hamscher et al., 1992; Dressler and Struss, 1996), which has strong relationships with the methods for fault diagnosis used in the engineering disciplines, such as control theory and statistical decision making (Cordier et al., 2004; Gertler, 1998; Gertler et al., 1995; Basseville and Nikiforov, 1993). Within consistency based diagnosis, a diagnosis is a set of components whose abnormal behaviors are a consistent explanation to why the system does not behave as intended. Further, a minimal diagnosis is a minimal such explanation. Considering consistency based diagnosis, fault isolation can be performed by computing a set of minimal diagnoses from the diagnostic test results.

Today, many embedded systems include multiple agents (Hayes, 1999; Leen and Heffernan, 2002; Navet et al., 2005; Hristu-Varsakelis

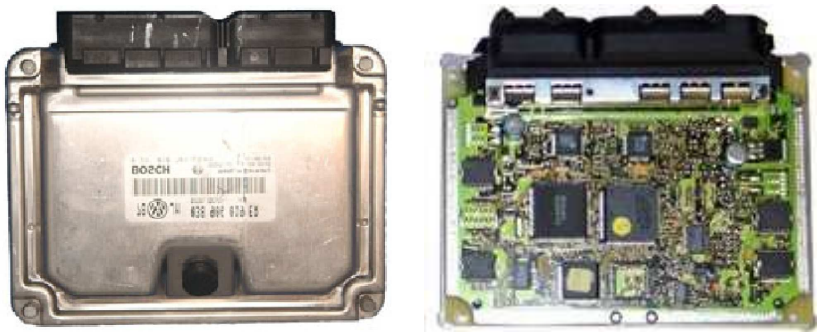



FIGURE 1.1: Outside and inside of an ECU by Bosch for an Audi personal vehicle. The ECU includes electronic components and software.

and Levine, 2005) for control and supervision. The agents can share for example sensor values over a network and the systems have therefore moved from being centralized to becoming distributed embedded systems. Centralized fault isolation can be performed based on all diagnostic test results in all agents, and when considering diagnoses, the result is a set of global diagnoses. The minimal global diagnoses are the minimal consistent explanations for the abnormal behavior of the complete system. Similarly, the diagnoses computed based on only the test results in one agent are denoted local diagnoses.

If each diagnostic system is independent of the other diagnostic systems then the results from the local fault isolations can be used directly since for example the sets of local minimal diagnoses in the agents will together directly form the set of minimal global diagnoses. However, a component such as a sensor component might be used by several agents, and the diagnostic system in one agent is therefore dependent on the diagnostic systems in the other agents. Considering diagnoses, the sets of local diagnoses in the agents are no longer guaranteed to form the set of global diagnoses, if the diagnostic systems are dependent. For such systems, it is therefore no longer possible to directly use the local fault isolations since the results are not guaranteed to be globally correct. How to perform local fault isolation in distributed embedded systems such that the results are globally correct is one of the topics of this thesis.

The background to the four papers presented in this thesis will be discussed below, and the publications relating to the thesis will be described. The next chapter will give an overview of the papers and for each paper state its contributions.

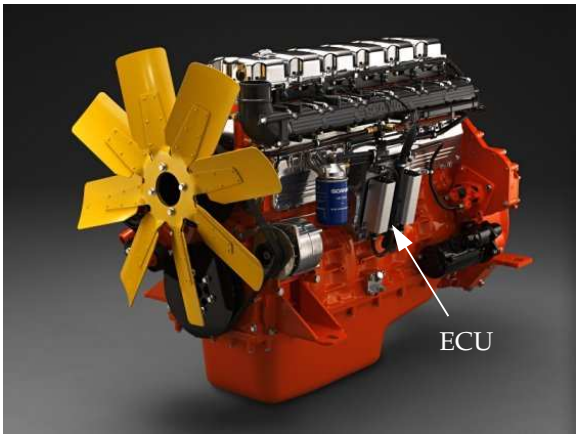



FIGURE 1.2: The figure shows a 12 liter industrial engine from Scania (Scania, 2007). The engine ECU is attached to the side of the engine and its objective is to supervise the engine for abnormal behavior and to control the engine such that emissions are minimized and performance maximized.

## 1.1 BACKGROUND TO THE THESIS

The work in this thesis is motivated by diagnostic systems used in automotive vehicles (Navet et al., 2005; Leen and Heffernan, 2002; Gertler, 1998; Hristu-Varsakelis and Levine, 2005; Struss and Price, 2003), and in particular that used in a Scania heavy duty vehicle. In automotive vehicles, the diagnostic systems are implemented in electronic control units (ECUs). Figure 1.1 shows for example an ECU for a personal vehicle from Audi while Figure 1.2 shows an ECU attached to a 12 liter industrial engine from Scania. The diagnostic system in the Scania ECU is responsible for the supervision of the components affecting the performance of the engine.

Diagnostic systems in automotive vehicles typically store a diagnostic trouble code (DTC) when a component has been detected to behave abnormally (SAE, 2003; Price, 1999; ISO, 1999). In for example personal vehicles following the OBD-II (On Board Diagnostic) standard, the DTCs can be read out with a standardized OBD-II reader, such as those shown in Figure 1.3. In the first generations of diagnostic systems used in automotive vehicles, each diagnostic test supervised exactly one component for abnormal behavior. Therefore, the DTCs could be used to state exactly which components that were behaving abnormally. Due to higher demands on fault diagnosis, such as reduced emission levels (EU, 2005; EPA, 2005), more components


FIGURE 1.3: Two different OBD-II code readers, one from Ford (left) and one from Actron (right) (Amazon, 2007). The readers are connected to the vehicles on-board diagnostic system and are used to collect the DTCs.

have to be supervised by the diagnostic systems. However, it is not possible to design one new diagnostic test for each additional component that should be supervised since the number of sensors is limited. Therefore, a trend in the automotive industry is the introduction of diagnostic tests that supervise several components at the same time, often denoted plausibility tests. Since there is no longer a one to one relationship between a test and an abnormal component, more elaborate fault isolation algorithms have to be used to be able to isolate the components that are behaving abnormally among all the components supervised by the tests. To perform fault isolation for plausibility tests is one of the motivations for the work presented in this thesis.

Fault isolation can be performed directly using a model of the complete system and a general diagnostic engine, such as the one presented in (Kleer and Williams, 1987) or similar algorithms. Using such algorithms, fault isolation can be performed by checking if the model, the observations, and the normal behavior of all components are consistent. If it is not consistent then it can be concluded that there is some fault present in the system and further checks can be performed to gain the global diagnoses. However, since the diagnostic tests used in automotive vehicles, and especially those in Scania heavy duty vehicles, have good performance, it is an advantage to base the more elaborate fault isolation on these diagnostic tests. Therefore, this thesis studies fault isolation based on diagnostic test results.

In addition to plausibility tests, another trend in automotive vehicles is the inclusion of multiple ECUs, which gives a distributed system. For example, Figure 1.4 shows part of the distributed system in

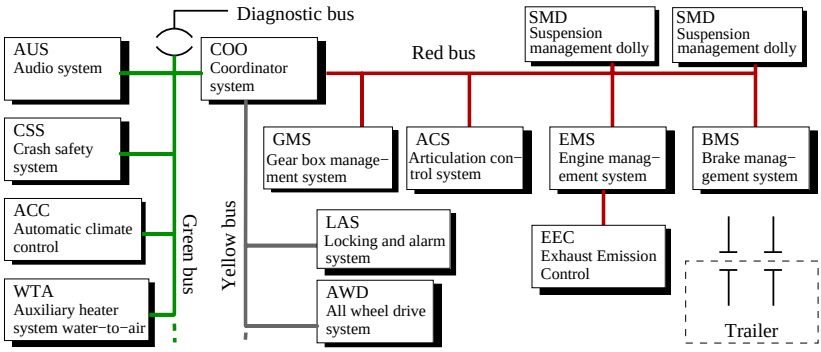



FIGURE 1.4: The distributed system in current Scania heavy-duty vehicles. The distributed system consists of dozens of ECUs, such as the CSS crash safety system that is responsible for protecting the driver and the passengers in case of a road traffic accident.

a Scania heavy duty truck and Figure 1.5 shows a typical distributed system in a personal automotive vehicle. In the first generations of distributed automotive systems, each ECU supervised a unique set of components and the diagnostic systems were therefore independent. However, due to the increased demands on diagnosis, the diagnostic systems have started to supervise components physically connected to and supervised by other ECUs. As described earlier, the local fault isolation is therefore no longer guaranteed to be globally correct.

## 1.2 PUBLICATIONS

This thesis includes research that has been presented in the following publications.

- Earlier versions of Paper I have been presented in peer reviewed publication (Biteus et al., 2005), in publication (Biteus et al., 2004b), and in the Licentiate thesis (Biteus, 2005). A shorter version of the paper has been accepted for publication as journal paper (Biteus et al., 2007).
- An earlier version of Paper II has been presented in peer reviewed publication (Biteus et al., 2006a). The paper is partially based on results presented in the Licentiate thesis (Biteus, 2005). A shorter version of the paper has been submitted to IEEE Transactions on Control Systems Technology.
- An earlier version of Paper III has been presented in peer reviewed publication (Biteus et al., 2006b). A shorter version of

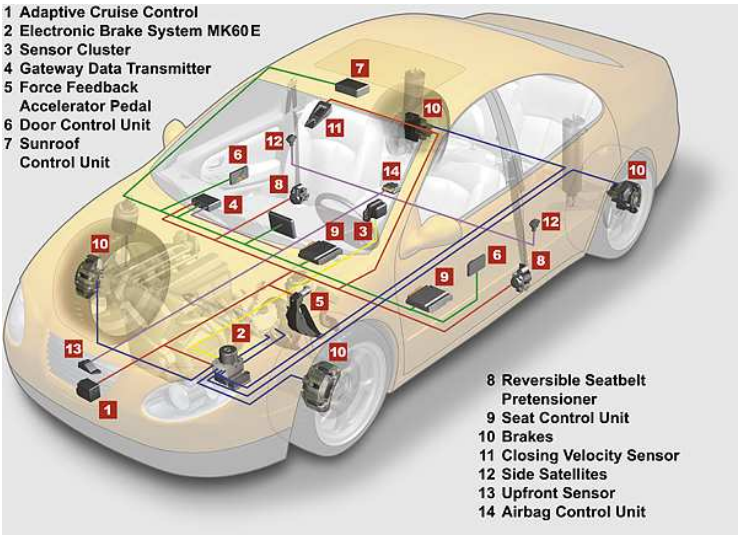



FIGURE 1.5: The distributed system in a personal automotive vehicle (AA1CAR, 2007). Multiple ECUs are distributed over the vehicle and are responsible for control and supervision of different parts of the vehicle, such as the adaptive cruise control or the airbag control unit.

the paper has been submitted to IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems.

- The research later published as (Biteus et al., 2006b) and on which Paper III is based has been applied for patent protection.
- Paper IV has been published as journal paper (Åslund et al., 2006). An earlier version of Paper IV has been presented in peer reviewed publication (Åslund et al., 2005). Related to the paper is the publication (Biteus et al., 2004a) that describes some relations between aircraft safety and fault diagnosis.

The following papers relate to diagnosis and have been produced under the Ph.D. thesis project but have not been included in this thesis.

- Part II of the Licentiate thesis (Biteus, 2005) has not been included in this thesis. The topic of Part II is simulation based residual generation. The residuals are constructed such that they can be used to construct diagnostic tests for embedded systems. Part II includes the peer-reviewed publication (Biteus and Nyberg, 2003) and the earlier publication (Biteus and Nyberg, 2002).

# Chapter 2

# Overview and Contributions of the Papers

This chapter will give an overview of the four papers presented in this thesis. To be able to describe the papers, the basic ideas within fault isolation will first be introduced. After the introduction, a summary of each paper will be given including its objectives and contributions.

## 2.1 Introduction to Fault Isolation

The overall objective in fault isolation is to compute a list of possibly abnormal components, which is ordered such that the components that most likely are abnormal are ranked high while those that are less likely are ranked low. The ranking can for example be used by a repair technician by improving troubleshooting. The components ranked high are first checked for abnormal behavior, and only after these have been checked, the other components are checked in descending rank.

### *2.1.1 Fault Isolation Directly Based on Test Results*

Considering diagnostic system based on diagnostic tests, the most direct approach to perform fault isolation is to directly present the diagnostic test results for the repair technician or the fault tolerant control

system and let the technician or the fault tolerant control system use these results to isolate the abnormal components. This is the common approach in automotive vehicles where the test results correspond to DTCs, see Section 1.1.

### 2.1.2 *Test Results and Diagnoses*

Another approach to perform fault isolation is to create a list of abnormal components based on the minimal diagnoses. Similar to components, it is an advantage to rank the diagnoses since the number of diagnoses might be high. The most probable diagnosis is ranked first and the other follow in descending order.

Consider for example the diagnostic system with the diagnostic tests $T_1$, $T_2$, and $T_3$, the components A, B, and C, and the following isolation structure.

| Test | A | B | C |
|---|---|---|---|
| $T_1$ | × | × | |
| $T_2$ | | × | × |
| $T_3$ | | × | |

In the isolation structure, a cross means that a test is sensitive to faults making the corresponding component to behave abnormally. Test $T_1$ is for example sensitive to faults in component A and B. Given that test $T_1$ and $T_2$ have responded, the conclusion is that component A or B is abnormal since $T_1$ has responded, and that component B or C is abnormal since $T_2$ has responded. One diagnosis is in this case that B is abnormal since this would explain both that test $T_1$ and $T_2$ have responded. Further, two more diagnoses exist, one is that both A and C are abnormal and another is that A, B, and C are abnormal. In set notation, these diagnoses are denoted $\{B\}$, $\{A, C\}$, and $\{A, B, C\}$, respectively. Out of these diagnoses, the diagnoses $\{B\}$ and $\{A, C\}$ are minimal since these are the minimal diagnoses, considering subsets, explaining the diagnostic test results.

A ranking of the components can be based on the minimal diagnoses by considering the cardinality, i.e. the size, of the diagnoses. For the minimal diagnoses $\{B\}$ and $\{A, C\}$, the following ranking of abnormal components can be created.

| Rank | Abnormal comp. |
|---|---|
| 1st | B |
| 2nd | A and C |

**Information in non responded tests**

Why is not the information that test $T_3$ has not responded used? If $T_3$ has not responded then it can be concluded that component B is

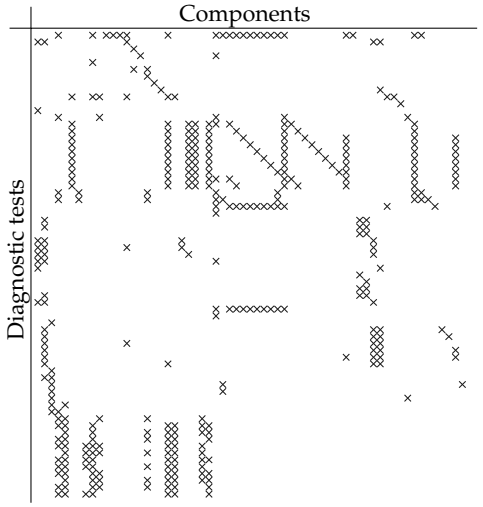



FIGURE 2.1: The isolation structure in the engine management system, which controls and supervises the engine in a Scania heavy duty truck.

behaving normally and component A and C are therefore abnormal since only these can explain the test results. This type of diagnosis reasoning is not considered in this thesis since it can lead to incorrect diagnoses. If for example test $T_3$ has a low probability to detect a fault in component B, then B might be abnormal even though $T_3$ has not responded. If the information that $T_3$ has not responded is used then B will incorrectly be assumed to be normal and A and C will incorrectly be stated to be abnormal. By not using the information of non responded tests, the possibility for the diagnostic system to isolate abnormal components is reduced, however it ensures that the diagnostic system does not give faulty diagnoses.

In automotive applications, the probabilities for false alarms are typically set low by choosing high enough thresholds for when the tests should respond. A consequence of the high thresholds is that, in some cases, the probability for a test to respond due to an abnormally behaving component is low. Therefore, it is not unlikely that the information about non responded tests will lead to faulty diagnoses.

**Ranking based on diagnoses with minimal cardinality**

For the simple example with three tests, the diagnoses can easily be computed by hand. However, this is not the case for larger systems, such as the diagnostic system in the engine management system (EMS) used in heavy duty trucks from Scania, see Figure 2.1. The figure

shows the isolation structure for the diagnostic system in the EMS, where each row corresponds to a test and each column corresponds to a component. To deal with such larger systems, algorithms exist that automatically compute the set of minimal diagnoses, see for example (Reiter, 1987; Hamscher et al., 1992).

Even though automatic algorithms exist, the complexity when computing the diagnoses is in worst case exponentially increasing in the number of tests, and it is therefore in some cases too computationally expensive to compute the complete set of minimal diagnoses. One way to reduce the computational burden is to only consider the diagnoses with minimal cardinality, i.e. the diagnoses that include a minimal number of components, for example the diagnosis $\{B\}$ in the example above. The ranking based on the minimal cardinality diagnoses would in the example be as follows.

| Rank | Abnormal comp. |
|---|---|
| 1st | B |

The focus on the diagnoses with minimal cardinality always removes the components ranked 2nd or lower, while the components ranked 1st are always kept.

### 2.1.3 *Fault Isolation for Distributed Systems*

In distributed systems, a local ranking is based on a set of local diagnoses in one agent, while the global ranking is based on the set of global diagnoses. From the discussion in Chapter 1 follows that if the diagnostic systems are dependent then the local rankings are not guaranteed to be globally correct. This is in contrast to the global ranking, which is globally correct.

Consider for example a system consisting of two agents that include diagnostic systems with one test each, such that the isolation structures for these systems are as follows.

Agent 1

| Test | A | B | C |
|---|---|---|---|
| $T_{11}$ | × | × | |

Agent 2

| Test | A | B | C |
|---|---|---|---|
| $T_{21}$ | | × | × |

Assume that both tests have responded, then the minimal local diagnoses in the first agent are $\{A\}$ and $\{B\}$, and the minimal local diagnoses in the second agent are $\{B\}$ and $\{C\}$. Therefore, the local rankings in the agents are as follows.

Agent 1

| Rank | Abnormal comp. |
|---|---|
| 1st | A or B |

Agent 2

| Rank | Abnormal comp. |
|---|---|
| 1st | B or C |

The local rankings are not globally correct, since if both tests are used to state a global ranking then component B would in both rankings be ranked first and A and C would be ranked second.

Automotive applications can consist of dozens of agents and include multiple components used by several agents, similar to component B in the example. Therefore, the computation of the global ranking requires access to the minimal global diagnoses. Further, since for example fault tolerant control can be performed locally in one agent, the minimal global diagnoses should be available in each agent.

Given this introduction to fault isolation, the four papers presented in this thesis will now be introduced.

## 2.2 PAPER I – AN ALGORITHM FOR COMPUTING THE DIAGNOSES WITH MINIMAL CARDINALITY IN A DISTRIBUTED SYSTEM

As discussed in the previous section, it is an advantage to have access to the global diagnoses in each agent and sometimes these global diagnoses are focused on to the set of global diagnoses with minimal cardinality. A centralized approach to compute the global diagnoses in a distributed system is to transmit all test results from all agents to a central agent responsible for diagnosis. However, a drawback with such an approach is that the worst case complexity when computing the minimal global diagnoses increases exponentially with the number of agents in the system, since the number of tests increases with the number of agents. A centralized approach therefore requires a powerful central diagnostic agent, and the system would not be robust against disconnections of this agent.

### *2.2.1 Objective*

The objective of Paper I is to design an algorithm that computes the set of minimal cardinality global diagnoses in a distributed cooperation between the agents in a distributed system. The algorithm should be designed such that the maximum computational load on any agent is minimized.

### *2.2.2 Summary*

To fulfill the objective, Paper I designs an algorithm whose main idea is to distribute parts of the computation of the global diagnoses to the different agents. Further, instead of computing the minimal cardinality global diagnoses, the algorithm computes independent sets of

diagnoses such that these directly represent the set of minimal cardinality global diagnoses.

**Module diagnoses**

Independent sets of diagnoses are in the paper denoted sets of module diagnoses. The sets are independent in the sense that a component included in a diagnosis in one set is not included in a diagnosis in any other set of module diagnoses. The objective of the designed algorithm is to compute the sets of minimal cardinality module diagnoses.

The module diagnoses will be exemplified for a system consisting of two agents that have the following isolation structures, and where both tests have responded.

Agent 1

| Test | A | B | C |
|---|---|---|---|
| $T_{11}$ | × | | |

Agent 2

| Test | A | B | C |
|---|---|---|---|
| $T_{21}$ | | × | × |

The minimal cardinality local diagnoses are $\{A\}$ in the first agent and $\{B\}$ and $\{C\}$ in the second agent. The minimal cardinality global diagnoses are $\{A, B\}$ and $\{A, C\}$. In this example, the sets of minimal cardinality local diagnoses are the sets of minimal cardinality module diagnoses, since the diagnosis in the first agent is independent of the diagnoses in the second agent.

If the sets of minimal cardinality local diagnoses are independent, then the objective is fulfilled since the sets of minimal cardinality module diagnoses are directly available. However, this is not always the case. This is for example not the case for the three agent system that has the following isolation structures and where all tests have responded, since the first and second agent both include the diagnosis $\{B\}$.

Agent 1

| Test | A | B | C | D |
|---|---|---|---|---|
| $T_{11}$ | × | × | | |

Agent 2

| Test | A | B | C | D |
|---|---|---|---|---|
| $T_{21}$ | | × | × | |

Agent 3

| Test | A | B | C | D |
|---|---|---|---|---|
| $T_{31}$ | | | | × |

If the sets of local diagnoses are dependent then the set of agents is instead partitioned into modules, where the sets of local diagnoses in one module are independent of the sets of local diagnoses in the other modules. In the three agent example, the set of agents is partitioned into one module consisting of the first and the second agent and another module consisting of the third agent. The minimal cardinality module diagnosis is $\{B\}$ for the first module and $\{D\}$ for the second module, and these directly represent the minimal cardinality global diagnosis $\{B, D\}$.

The main advantage when computing the minimal cardinality module diagnoses instead of the minimal cardinality global diagnoses, is

that the maximum load on any agent is reduced from exponentially to linearly increasing, considering the number of modules in the system.

After the set of agents in a system has been partitioned into modules based on the current test results, the objective is to compute the sets of minimal cardinality module diagnoses. The algorithm designed in Paper I lets each agent compute its own set of minimal local diagnoses, and then the sets of minimal cardinality module diagnoses are computed in a distributed cooperation between the agents in each module. The distribution further reduces the maximum load on any agent from linearly increasing to become constant for systems larger than a certain number of agents.

#### Evaluation of the designed algorithm

For a set of systems studied in the paper, where each system includes four agents, there is in mean a 40 % reduction in maximum processor load on any agent compared to a centralized algorithm. Further, the gain increases for larger systems, and when the systems for example include eight agents then there is in mean a reduction of over 99.9 %. If the centralized algorithm used in the comparison computes the minimal cardinality module diagnoses instead of the minimal cardinality global diagnoses, then the reduction is still over 70 % when using the designed algorithm for the systems with eight agents. This shows that for these systems the distribution of computations reduces the maximum load with 70 % while the partition of the agents into modules further reduces the maximum load to a total of over 99.9 %.

The reduction in computational load comes with the drawback that the load on the network increases due to the cooperation between the agents. For the studied sets of systems, the number of transmissions increases from about 50 to about 1 500 for systems with four agents, while for eight agents it increases from about 110 to about 3 000 transmissions. An important conclusion from the evaluation is that both the gain in processor load and the cost in transmissions for the designed algorithm increase linearly with the number of modules.

### *2.2.3 Contributions*

In summary, the contributions of Paper I are as follows.

- The algorithm that efficiently computes the sets of minimal cardinality module diagnoses without first computing the set of minimal module diagnoses. The sets of minimal cardinality module diagnoses are a direct representation of the minimal cardinality global diagnoses.

- The strategy of partitioning a system into sub-systems with respect to the test results such that the complexity when performing global fault isolation reduces from exponentially to linearly increasing in the number of modules.

## 2.3 PAPER II – DISTRIBUTED DIAGNOSIS USING A CONDENSED REPRESENTATION OF DIAGNOSES WITH APPLICATION TO AN AUTOMOTIVE VEHICLE

A drawback when using the global diagnoses for repair of an agent or for fault tolerant control in an agent is that they include many components that are not used by the agent, and thereby do not affect the behavior of the agent. The unaffecting components make the number of global diagnoses to be unnecessary high and each global diagnosis unnecessary large. For example, in the automotive application from Scania, the EMS (engine management system) does not use the catalytic converter component and it is therefore unnecessary to include the component in the global diagnoses when they are used in the EMS.

### 2.3.1 *Objective*

The main objective of Paper II is to develop a method that makes a representation of the global diagnoses available in each agent, which does not include unaffecting components. Further, when the representation is computed, the maximum computational load on any agent should be reduced as much as possible.

### 2.3.2 *Summary*

To fulfill the objective, a novel type of diagnosis, condensed diagnosis, is defined in Paper II. Each agent has a unique set of minimal condensed diagnoses, where each condensed diagnosis only includes components affecting the agent but preserves the cardinality of the global diagnoses.

Assume for example that $\{A, B, C, D\}$ is a global diagnosis, where only component $A$ affects the agent. A condensed diagnosis representing this global diagnosis is a tuple $\langle\{A\}, 3\rangle$, where $A$ is the component the agent uses, and the number 3 represents the unaffecting components $B$, $C$, and $D$ such that the cardinality of the global diagnosis is preserved. The cardinality of the condensed diagnosis $\langle\{A\}, 3\rangle$ is $4 = |\{A\}| + 3$ and this is equal to the cardinality of the global diagnosis.

The condensed diagnoses are globally correct since they preserve the cardinality of the global diagnoses. Consider for example a system where the global diagnoses are $\{B\}$ and $\{A, C\}$ and where the first agent only is affected by the behavior of component A and B. The condensed diagnoses are in the first agent $\langle\{B\}, 0\rangle$ and $\langle\{A\}, 1\rangle$ with cardinality one and two respectively. Based on the condensed diagnoses, the following ranking can be created.

| Rank | Abnormal comp. |
|---|---|
| 1st | B |
| 2nd | A |

Since the condensed diagnoses are globally correct, the ranking coincides with the global ranking.

Due to the removal of the unaffecting components, each condensed diagnosis represents one or several global diagnoses and the number of minimal condensed diagnoses in each agent is therefore reduced compared to the minimal global diagnoses. Due to this reduction in number of diagnoses, the minimal condensed diagnoses are suitable to use in fault tolerant control or for repair of the agent since fewer diagnoses have to be checked before the agent is repaired or controlled with respect to the abnormal components.

To reduce the maximum computational load on any agent when the condensed diagnoses are computed, an algorithm is designed that in a distributed cooperation between the agents computes the sets of minimal condensed diagnoses in each agent.

**Evaluation on an automotive vehicle**

To evaluate the condensed diagnoses and the designed algorithm, an application study is in the paper performed on a part of the distributed system used in a heavy duty vehicle from Scania. One of the engines for the vehicles that use the studied distributed system is shown in Figure 2.2. The engine fulfills the Euro 5 emission restrictions that will be enforced in Europe in 2009 (EU, 2005).

The part of the distributed system that is studied here includes three agents: the EMS (engine management system), which controls and supervises the engine; the selective catalytic reduction system (SCR), which controls and supervises the after-treatment system including for example a catalytic converter; and the coordinator system (COO), which has a coordinating functionality. The three agents supervise a total of 85 components using over 100 diagnostic tests. Figure 2.3 shows a schematic overview of the system where a component supervised by an agent is connected with a line. Three signals from components are transferred over the network to the other agents.


FIGURE 2.2: The figure shows a 16 liter 500 hp heavy duty truck engine from Scania (Scania, 2007). The engine management system (EMS) is attached to the upper part of the engine. This engine fulfills the Euro 5 emission restrictions using among other things a selective catalytic reduction system (SCR) for after-treatment of the exhaust gases.

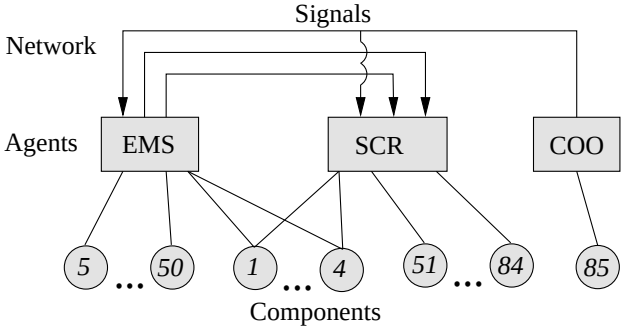



FIGURE 2.3: A schematic overview of the distributed system used in the Scania application.

The evaluation has shown that if two abnormal components are present in the system, then the number of minimal condensed diagnoses is in mean reduced by 70 % compared to the number of minimal global diagnoses. Further, the reduction in the number of diagnoses increases with the number of abnormal components and reaches for example a 90 % reduction for four abnormal components.

Compared to a centralized algorithm that computes the set of minimal global diagnoses, the designed algorithm gives a mean reduction of the maximum processor load on any agent with 50 and 85 % when the automotive system includes two and four abnormal components respectively. Further, the reduction in processor load continues to increase with the number of abnormal components. In contrast to Paper I, there is no significant increase in the number of needed transmissions.

#### Condensed diagnoses with minimal cardinality

In addition to the computation of the sets of minimal condensed diagnoses, Paper II also includes an algorithm that computes the set of minimal cardinality condensed diagnoses in each agent. Also this extended algorithm is applied to the automotive vehicle with results as those described above.

### *2.3.3 Contributions*

In summary, the contributions of Paper II are as follows.

- The method that makes a representation of the global diagnoses available in each agent, which only includes components affecting the behavior of the agent.
- The novel concept of condensed diagnosis. The condensed diagnoses in one agent preserve the cardinality of the global diagnoses while excluding all unaffecting components.
- The algorithm that in a distributed cooperation between the agents computes the set of minimal or minimal cardinality condensed diagnoses in each agent, without first computing the set of minimal or minimal cardinality global diagnoses respectively.
- The application of the condensed diagnoses to the diagnostic system in an automotive vehicle.

## 2.4 PAPER III – DETERMINING THE FAULT STATUS OF A COMPONENT AND ITS READINESS, WITH A DISTRIBUTED AUTOMOTIVE APPLICATION

As described in Section 1.1, the introduction of plausibility tests removes the direct relation between abnormal components and tests, since components might exist that are only suspected to be abnormal. Further, if the additional evaluation of tests can not improve the fault isolation of a component then the component is ready. The introduction of plausibility tests also removes the direct relation between diagnostic tests and ready components, since a component might be ready even though all tests supervising the component have not been evaluated. It is an advantage to get readiness for all components, which can be achieved by evaluating all tests. However, this approach is in for example automotive applications not always feasible due to for example limited processing power.

### 2.4.1 *Objective*

Considering plausibility tests, one objective of Paper III is to decide if a component is ready, and if it is not ready to decide which tests that should be evaluated to gain readiness. Another objective is to develop a strategy that gives readiness for as many components as possible with as few evaluations of diagnostic tests as possible. Further, since the focus of this thesis is on fault diagnosis for embedded distributed systems, the methods designed in the paper should be applicable for both centralized and distributed systems.

### 2.4.2 *Summary*

In the paper, the fault status of a component is defined, which can be either faulty, suspected, or normal. If a test has responded that only supervise one component then the fault status of the component is faulty. If the component is supervised only by responded plausibility tests then the fault status of the component is suspected. Otherwise, the fault status of the component is normal.

Consider for example a system with four diagnostic tests that have the following isolation structure.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $T_1$ | × | × | | | |
| $T_2$ | | × | × | | |
| $T_3$ | | | | × | |
| $T_4$ | | × | | | |

If tests $T_1$, $T_2$, and $T_3$ have responded, and test $T_4$ has not yet been evaluated, then the fault status of component D is faulty since $T_3$ is a single component test. It is not known if it is component A, B, or C that have caused test $T_1$ and $T_2$ to respond since they are plausibility tests, therefore the fault statuses of component A, B, and C are suspected. Further, there is no indication in the test results that component E is behaving abnormally, and its fault status is therefore normal. The following ranking can be computed based on the fault statuses.

| Fault status | Rank | Abnormal comp. |
|---|---|---|
| Faulty | 1st | D |
| Suspected | 2nd | A or B or C |
| Normal | 3rd | E |

**Readiness**

Considering plausibility tests, the paper clarifies when the fault status of a component is ready given the information about which tests that have been evaluated, which that have responded, and which that could be evaluated in the future. For the example above with four tests, it can be computed that component D is ready, while components A, B, and C are not ready. Components A, B, and C are not ready since if test $T_4$ is evaluated and responds, then the fault status of B changes to faulty and the fault status of both A and C change to normal. Using the clarified relations between tests, fault status, and readiness, an efficient algorithm is designed that computes the fault status and readiness for all components.

**Evaluation of the fault status and the readiness**

The algorithm designed in the paper is applied to both the diagnostic system in the EMS and in the SCR used in a heavy duty vehicle from Scania. The diagnostic systems are somewhat more complex than those studied in Paper II and include both more diagnostic tests and components. In this paper, both the diagnostic system in the EMS and the SCR consist of about 70 diagnostic tests that supervise about 50 and 60 components respectively.

Compared to an algorithm that directly computes the fault status and readiness for all components, the designed algorithm gives a reduction of the processor load with 80 to 90 % for the EMS and the SCR.

**Readiness and meaningful tests**

If a component is not ready, then it is interesting to know which diagnostic tests that should be evaluated to gain readiness, and these tests

are denoted meaningful. The paper exactly states which tests that are meaningful for a given set of not ready components.

Given a set of meaningful tests, a strategy is in the paper designed that computes which meaningful test that will give the most number of ready components. For the automotive vehicle affected by two abnormal components, the best test will for example in mean give 1.7 new ready components, while the next best test will in mean only give 1.0 ready components. By evaluating the meaningful tests in the best order, the number of tests that has to be evaluated is reduced to a minimum, and thereby reduces for example processor usage.

#### Extension to distributed systems

Considering distributed systems, the local fault status and readiness of a component is computed in each agent, while the global fault status and readiness is computed for the complete system. In the paper, the relations between the local and global fault statuses and readiness are clarified. Using the relations, it is for example possible to compute the global readiness of a component based on the local fault statuses and local readiness computed in the different agents.

#### Fault status and diagnoses

The fault status differs from the diagnoses used in Paper I and II in that the fault statuses give the components that certainly are abnormal, the components that might be abnormal, and the components that are normal, while each diagnosis is a set of components where the components abnormal behaviors are consistent with the test results. A drawback when using the fault status compared to the diagnoses is that the ranking is not as good as when the minimal diagnoses are used. As an example, for the responded tests $T_1$, $T_2$, and $T_3$, the minimal diagnoses $\{B, D\}$ and $\{A, C, D\}$ can be calculated. It can be seen that, based on the minimal diagnoses, component B will be ranked 1st together with component D. This is in contrast to the ranking based on the fault status where only D is ranked first.

The advantage of the fault status and readiness is that the complexity of its computation is linearly increasing in the number of tests while the computation of the diagnoses is exponentially increasing. This lower complexity makes it for example possible to compute the fault status and readiness for the automotive application even when it is practically intractable to compute the set of minimal diagnoses.

### 2.4.3 *Contributions*

In summary, the contributions of Paper III are as follows.

- The propositions describing the relations between diagnostic plausibility tests, diagnoses, fault status, and readiness, for both centralized and distributed systems.
- The strategy for scheduling the evaluation of meaningful diagnostic tests such that the evaluation of the tests fastest leads to most ready components.
- The algorithms that efficiently, compared to a direct implementation, computes the fault status and readiness of all components and the meaningful diagnostic tests.
- The application of the designed algorithms to the diagnostic system in an automotive vehicle.

## 2.5 PAPER IV – SAFETY ANALYSIS OF AUTONOMOUS SYSTEMS BY EXTENDED FAULT TREE ANALYSIS

In contrast to the other papers, which discuss *how* fault diagnosis can be performed, Paper IV discusses the *effect* that fault diagnosis has on safety.

One approach to increase safety is to let embedded software perform autonomous decisions to avoid dangerous situations, and a key mechanism is the use of fault tolerant control based on fault diagnosis. Decisions that previously were taken by a pilot or a driver, can now be taken autonomously by the fault tolerant control system. To be able to analyze if a system is safe or not, a common approach is to use fault tree analysis (FTA). Therefore, a natural question in many modern systems that include sub-systems like fault diagnosis, fault-tolerant control, and autonomous functions, is how to include the performance of these algorithms in a fault tree analysis for safety.

### 2.5.1 *Objective*

To develop a method that makes it possible to include the performance of fault diagnosis algorithms in fault tree analysis for safety, and to investigate the relation between requirements on fault diagnosis performance and requirements on system safety.

### 2.5.2 *Summary*

A systematic way to include fault diagnosis in fault tree analysis is proposed in Paper IV. It is shown both how safety can be analyzed

and how the interplay between fault diagnosis algorithm design in terms of missed detection rate and false alarm rate is included in the fault tree analysis. Examples illustrate analysis of diagnosis system requirement specification and algorithm tuning. As mentioned in Section 2.1.2, the false alarm rate and the missed detection rate is the link to parameter setting of fault diagnosis algorithms, and is thus the foundation for both requirements on system safety on one hand and for fault diagnosis algorithm tuning on the other hand.

### 2.5.3 *Contributions*

In summary, the contributions of Paper IV are as follows.

- The systematic way of introducing fault diagnosis in fault tree analysis.
- The generic example illustrating how to transfer requirements on system safety to performance requirements on the fault diagnosis algorithms, and the illustration of an optimization criterion useful for optimizing the parameters in the algorithms.

*Background Theory*

# Consistency Based Diagnosis

This chapter will briefly describe the concept of consistency based diagnosis. The motivation is not to give a complete introduction, 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 collections (Hamscher et al., 1992; Dressler and Struss, 1996).

## Consistency Based Diagnosis

A system consists of a set of components $\mathcal{C}$, which should be supervised by the diagnostic system. A component is something that can be diagnosed, such as pipes, sensors, and actuators. The objective of the diagnostic system is to detect and isolate the components that are behaving abnormally.

Model based diagnosis compares a model of a system with available observations. Deviations between the model and the observations can then be used to draw conclusions of the fault state of the system. A component can be in one or several behavioral modes where each mode describes the behavior of the component using a model. The objective in consistency based diagnosis is to derive a set of behavioral mode assignments to the components in the model, such that the model, the observations, and the behavioral mode assignments are consistent with each other.

### *Model, Observation, and Behavioral Modes*

A system is described by its system description, i.e. its model, here denoted SD. The system description consists of a set of logical rules, such as differential equations, system variable inequalities, etc. Similarly, the observations OBS consist of a set of logical rules, such as observed values of variables. A component can be in one of several different behavioral modes, and for each mode the behavior of the component is described by a model. Typically, each component $c \in \mathcal{C}$ has an abnormal mode AB, which does not have a model, a normal mode $\neg$AB, and one or several specific fault modes. The notation $AB(c)$ will be used when a component $c \in \mathcal{C}$ is in the abnormal mode.

It is sometimes preferable to only consider the AB and the $\neg$AB mode where the AB mode does not have a model. Some of the reasons for this are that this reduces the number of behavioral modes that has to be considered, and that only the normal behavior of component has to be modeled. Therefore, from now on the following assumption is made.

ASSUMPTION 1: *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 (Kleer et al., 1992). With this assumption, the notation in for example GDE (Hamscher et al., 1992) can be employed. This notation replaces the logical expressions with sets, where the sets are used to represent both conflicts and diagnoses.

EXAMPLE 1: If two components A and B are in the abnormal mode, this is written in logic form as $AB(A) \wedge AB(B)$ and can be represented by $\{A, B\}$ in the set notation. ◇

### *Diagnosis*

A diagnosis is in general terms an explanation of the behavior of a system. In consistency based diagnosis, the following definition of diagnosis is often used.

DEFINITION 1 (Diagnosis (Kleer et al., 1992)): *A diagnosis is a set of components* $D \subseteq \mathcal{C}$ *such that*

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

*is consistent.*

A diagnosis states a system mode consistent with the system description and the observations. Given the no fault mode assumption in Assumption 1, a superset of a diagnosis is also a diagnosis and this

leads to the notation minimal diagnoses. A diagnosis D is a minimal diagnosis if there is no proper subset $D' \subset D$ where $D'$ is a diagnosis. Under Assumption 1, 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, only minimal diagnoses are needed.

EXAMPLE 2: Consider a system with the sensors, denoted component A and B, and the following system description SD.

$$\neg AB(A) \rightarrow y_A = x$$
$$\neg AB(B) \rightarrow y_B = x$$

The system description states that if the component A is not abnormal then the sensor $y_A$ will equal the variable x, and if the component B is not abnormal then the sensor $y_B$ will equal the variable x. Assume that the following observations OBS have been done.

$$y_A = 1 \qquad\qquad y_B = 2$$

Consider now the proposed diagnosis $D = \{A\}$, for which

$$(\neg AB(A) \rightarrow y_A = x) \wedge (\neg AB(B) \rightarrow y_B = x) \wedge$$
$$(y_A = 1) \wedge (y_B = 2) \wedge AB(A) \wedge \neg AB(B)$$

is consistent, which shows that D is a diagnosis. Performing the same consistency check for all different sets of components give the diagnoses $\{A\}$, $\{B\}$, and $\{A, B\}$. Notice that the empty set $\emptyset$ is not a diagnosis. In the set, the minimal diagnoses are $\{A\}$ and $\{B\}$. ◇

## *Conflict*

In some diagnostic systems, for example automotive systems, the diagnoses are not obtained directly from the model and the observations. The diagnoses are in these systems instead computed from the set of conflicts, where a conflict typically is generated when a diagnostic test responds.

DEFINITION 2 (Conflict (Kleer et al., 1992)): *A conflict is a set of components $\pi \subseteq \mathcal{C}$ such 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 that is inconsistent with the observations and the model. A

set of conflicts is denoted $\Pi$. From the conflicts follow the minimal conflicts. A conflict $\pi$ is a minimal conflict if there is no proper subset $\pi' \subset \pi$ where $\pi'$ is a conflict. Similar to diagnoses, under Assumption 1 the set of minimal conflicts completely characterizes all possible conflicts.

EXAMPLE 3: Continuation of Example 2. Given the system description and the observations, it can be found that the conflict $\pi = \{A, B\}$ exists in the system. This means that the SD, the OBS, and the mode assignments

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

are inconsistent. Meaning that component $A$ and $B$ can not both be in the not abnormal mode. ◇

The relation between conflicts and diagnostic tests will be further discussed after the relation between conflicts and diagnoses has been described.

### *Relation between Conflicts and Diagnoses*

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 (Kleer, 1991) in a different notation.

THEOREM 1 (Conflicts to diagnoses): *Let $\Pi$ be the set of conflicts. The set $D \subseteq \mathcal{C}$ is a diagnosis if and only if*

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

*for all $\pi \in \Pi$.*

A diagnosis can be seen as special case of a hitting set, which is also denoted vertex cover.

DEFINITION 3 (Hitting set): *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 $F \in \mathcal{F}$.*

Similar to diagnoses and conflicts, a hitting set $S$ for the set $\mathcal{F}$ is a minimal hitting set if there is no proper subset $S' \subset S$ 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, compare with Theorem 1. It is also the case that the minimal diagnoses are the minimal hitting sets for the set of minimal conflicts. Notice that Assumption 1 has to be true for these

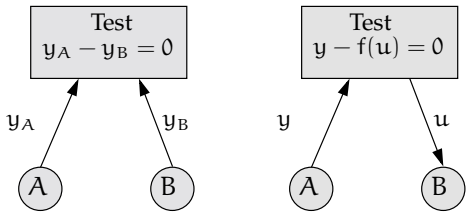



FIGURE 2.4: The figure shows two different tests that supervise components. The left test is described in Example 5 and the right is described in Example 6.

relationships to hold. In (Kleer et al., 1992), a proof for these relations is given. Due to these relations, algorithms for computing minimal hitting sets, such as (Wotawa, 2001), can be used when computing the minimal diagnoses.

EXAMPLE 4: Continuation of Example 3. Assume that the conflict $\pi = \{A, B\}$ has been detected by the diagnostic system. From this conflict the minimal diagnoses $\{A\}$ and $\{B\}$ can be calculated. ◇

## DIAGNOSTIC TESTS AND CONFLICTS

The evaluation of diagnostic tests is a common approach used to detect and isolate faults in a system. These tests might for example compare the value of a sensor with some prediction of the value of the sensor, and if these values fundamentally deviate from each other, it is concluded that some component or components are behaving abnormally in the system. This type of comparison between a sensor value and a predicted value is analytical redundancy relation and have been deeply studied within the fault detection and isolation (FDI) field, see for example (Gertler, 1998). In consistency based diagnosis, the results from the tests are stated as conflicts. The two examples below will illustrate the relation between tests and conflicts.

EXAMPLE 5: Consider a system including the two sensor components A and B, which measure the same temperature, $y_A = x$ and $y_B = x$. If the values of sensor A and sensor B fundamentally deviate, then both these sensors can not be behaving normally. The conflict is $AB(A) \wedge AB(B)$, which can be written as $\{A, B\}$ in set notation. One such test could be to calculate $y_A - y_B$ and if this value fundamentally deviates from zero then either A, B, or both are in the abnormal mode. The example is schematically shown as the left test in Figure 2.4. ◇

EXAMPLE 6: Consider now a system including a component A controlled by the actuator signal u and a sensor B with value y, see the right test in Figure 2.4. 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 $y - f(u)$ is a small value, i.e. if the model and the observations are consistent. If these are not consistent, then a conflict $\pi = \{A, B\}$ once again exists in the system. ◇

The design of tests demands expert domain knowledge and a good insight into diagnostic systems. See for example (Gertler, 1998; Chen and Patton, 1999; Patton et al., 2000; Nyberg, 1999a; Frisk, 2001; Krysander, 2006).

## MINIMAL CARDINALITY DIAGNOSES

In some applications, the set of minimal diagnoses is focused on to some smaller set of diagnoses, such as the most probable diagnoses or the diagnoses with minimal cardinality (Tuhrim et al., 1991), where the cardinality is the number of abnormal behaving components in a diagnosis.

DEFINITION 4 (Minimal cardinality diagnoses): *Let $\mathbb{D}$ be a set of diagnoses, then the set of minimal cardinality diagnoses is the set*

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

The set of minimal cardinality diagnoses includes only those diagnoses that include the fewest number of components. When considering repair, it is often natural to start the repair by checking the components included in the diagnoses with the fewest number of components, and then the other diagnoses are checked in increasing number of components. If for example the two diagnoses $\{A\}$ and $\{B, C\}$ have been detected. Then, considering only the number of components, component A should first be checked and if this has been found to be normal, the components B and C are checked.

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 the diagnoses should be computed in for example embedded distributed systems.

# *Papers*

# *Paper I*

# AN ALGORITHM FOR COMPUTING THE DIAGNOSES WITH MINIMAL CARDINALITY IN A DISTRIBUTED SYSTEM[1]

**Jonas Biteus**[⋆]**, Mattias Nyberg**[†]**, and Erik Frisk**[⋆]

[⋆] *Dep. of Electrical Engineering, Linköpings universitet,* SE-*581 83 Linköping, Sweden.* {biteus,frisk}@isy.liu.se.
[†] *Power-train division, Scania,* SE-*151 87 Södertälje, Sweden.* mattias.nyberg@scania.com

## ABSTRACT

In fault diagnosis, the set of minimal diagnoses is commonly calculated. However, due to for example limited computation resources, the search for the set of minimal diagnoses is in some applications focused on to the smaller set of diagnoses with minimal cardinality. The key contribution of the present paper is an algorithm that calculates the diagnoses with minimal cardinality in a distributed system. The algorithm is constructed such that the computationally intensive tasks are distributed to the different units in the distributed system, and thereby reduces the need for a powerful central diagnostic unit.

[1] A shorter version of this paper has been accepted for publication as (Biteus et al., 2007).

# 1 INTRODUCTION

Fault diagnosis is becoming more common in many applications, and one of the most widespread approaches for diagnosis is the consistency based diagnosis approach developed within the AI field, see (Kleer and Kurien, 2003) for an overview. In this approach, a diagnosis is a set of components whose abnormal behavior is a possible explanation to why a system does not behave as intended, and a minimal diagnosis is a minimal set of such components. Sometimes, it is computationally intractable to compute the complete set of minimal diagnoses. Therefore, focusing is used to reduce the search to some smaller set, for example the most probable diagnoses (Kleer, 1991) or the diagnoses with minimal cardinality (Tuhrim et al., 1991).

The key contribution of the present paper is a method that calculates the diagnoses with minimal cardinality in a distributed system. A distributed system consists of a set of agents, where an agent is a more or less independent software entity (Weiss, 1999). The diagnoses can, in distributed systems, be divided into two different levels, global diagnoses that are diagnoses for the complete distributed system and local diagnoses that are diagnoses for a single agent. The method designed in this paper first calculates the set of minimal local diagnoses in each agent. These sets of minimal local diagnoses are then used to calculate the set of global diagnoses with minimal cardinality.

Our work has been inspired by diagnosis in distributed embedded systems that are used in automotive vehicles (Gertler, 1998; Hristu-Varsakelis and Levine, 2005; Struss and Price, 2003) and especially that in a heavy duty vehicle from Scania. These systems typically consist of precomputed diagnostic tests that are evaluated in the different agents, which in the automotive industry correspond to electronic control units (ECUs) (SAE, 2003). The results from the diagnostic tests can be used to calculate the sets of local diagnoses in the agents. These embedded distributed systems typically consist of ECUs with both limited processing power and limited RAM memory. Therefore, the method designed in this paper calculates the global diagnoses with minimal cardinality in a cooperation between the agents, such that the computational expensive tasks are distributed between the different agents.

## *1.1 Related Work*

Model based diagnosis has been studied within several different fields, for example: (*i*) fault detection and isolation (FDI) methods (Gertler, 1998); (*ii*) statistical methods (Basseville and Nikiforov, 1993); (*iii*) discrete event systems where models can be described as some type of automata (Sampath et al., 1995); and (*iv*) AI methods (Kleer and Kurien,

2003; Reiter, 1987). The method developed in this paper is mostly related to the AI field.

In all four fields, most research has been aimed at the centralized diagnosis problem. These centralized methods can also be used for distributed systems by letting a single diagnostic agent collect data from all agents and then calculate the minimal global diagnoses. However, it is not always suitable to use a dedicated single diagnostic agent due to for example limited computing resources in each agent, robustness against agent disconnection, and scalability.

For distributed systems, algorithms exist, for example those designed in (Provan, 2002) and (Kurien et al., 2002), which are aimed at computing the minimal global diagnoses in a cooperation between agents. The methods in these papers differ from the method designed here in that they calculate all minimal global diagnoses. Algorithms also exist where the agents update the sets of signals transmitted between the agents, such that of minimal local diagnoses in each agent are consistent with the global set of signals, see for example (Roos et al., 2003). These methods differ from the method designed here in that they do not compute the set of global diagnoses, but only update the local diagnoses. It is straightforward to compute the set of minimal cardinality global diagnoses from the set of minimal global diagnoses. However, if only the set of global diagnoses with minimal cardinality is searched for, then the methods above would require the unnecessary computation of many global diagnoses. This in contrast to the method designed in this paper that directly calculates the global diagnoses with minimal cardinality.

**Focusing the diagnoses**

In this paper, the set of diagnoses is focused on to the diagnoses with minimal cardinality. An alternative is to focus set of diagnoses on to the set of most probable diagnoses. In for example (Kleer, 1991), the set of most probable diagnoses is computed using a centralized method. To use a focusing to the set of most probable diagnoses instead of the set of minimal cardinality diagnoses would give better results. However, it is computationally more difficult to compute this set and it is in many applications difficult, or practically impossible, to get good a-priori probabilities for the failure of the components. It can therefore be difficult to use the focusing to the most probable diagnoses.

As an alternative to compute the complete set minimal diagnoses, algorithms exist, such as the centralized algorithm presented in (Kleer and Williams, 1989), which focus the set of diagnoses by removing the less likely diagnoses while the diagnoses are computed. Typically is a limit set on the maximum number of diagnoses that is computed, if the number of diagnoses exceeds this number while the set of diagnoses is

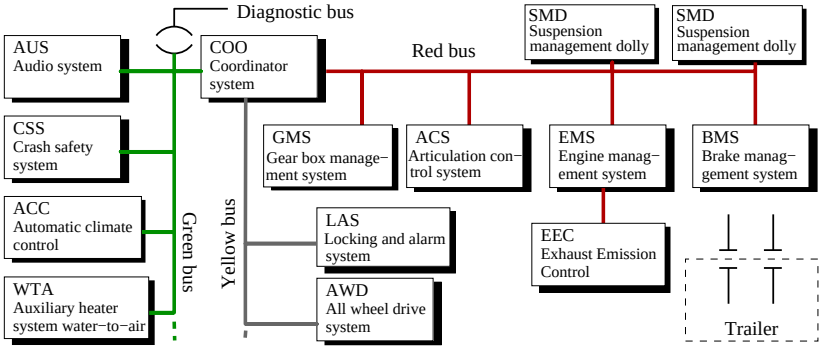



FIGURE 1: The distributed system in current Scania vehicles.

computed then the less likely of these diagnoses are removed such that the limit is not exceeded. The removal of diagnoses can be based on for example the cardinalities or the probabilities of the diagnoses. This type of removal reduces the computational complexity, however, by removing diagnoses, it is not guaranteed that the final set of diagnoses includes all diagnoses with for example minimal cardinality or highest probability. This is a difference compared to the method designed in this paper that aims at computing a complete set of diagnoses.

## 2 DISTRIBUTED CONSISTENCY BASED DIAGNOSIS

As an example of a distributed system, one configuration of the embedded system used in the heavy duty vehicles from Scania is studied, see Figure 1. In this system, there can be up to about 30 ECUs and roughly between 4 and 110 components are supervised by each ECU. Since the system consists of multiple ECUs, it would be an advantage if the different computation capacities in the different ECUs could be utilized, i.e. a distributed system is preferred.

Inspired by the system described above, a framework useful for distributed diagnosis is here designed. In this framework, a system consists of a set of components $\mathcal{C}$, which should be supervised by the diagnostic systems implemented in a set of agents $\mathcal{A}$. A component is something that can be diagnosed, such as sensors, actuators, cables, pipes, etc., and the components can be supervised by one or several agents.

EXAMPLE 1: Figure 2 shows a typical layout of agents and components. The system consists of two agents, a network, and four sensor

components, i.e. $S_1$ to $S_4$. The sensors $S_1$ and $S_2$ are physically connected to agent $A_1$, while the sensors $S_3$ and $S_4$ are physically connected to $A_2$. A diagnostic test checks the consistent behavior of the sensors connected to it with dashed lines. The diagnostic test in agent $A_1$ collects the value of sensor $S_3$ over the network, and uses this to check the consistency of the sensors $S_1$, $S_2$, and $S_3$. ◇

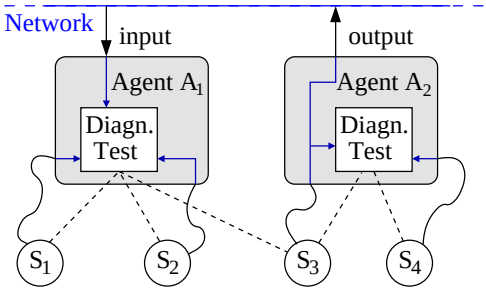



FIGURE 2: A typical layout of agents, network, components, and tests.

## 2.1 *Diagnoses and Conflicts*

To reduce the complexity, it is sometimes preferable to only consider the abnormal and the not abnormal mode, where the abnormal mode does not have a model. In this paper, only these two modes without fault models are considered and the notation from GDE (Kleer and Williams, 1987) will be used.

A diagnosis is a set of components $D \subseteq \mathcal{C}$, such that the abnormal behaviors of the components, the normal behaviors of the remaining components, the system description, and the observations are consistent (Kleer et al., 1992). A diagnosis D is a minimal diagnosis if there is no proper subset $D' \subset D$ where $D'$ is a diagnosis. Further, given the set of diagnoses $\mathbb{D}$, the set of minimal cardinality diagnoses is the set $\{D \in \mathbb{D} : |D| = \min_{D \in \mathbb{D}} |D|\}$.

A conflict is generated if a diagnostic test has responded. A conflict is a set of components $\pi \subseteq \mathcal{C}$, such that the normal behaviors of the components, the system description, and the observations are inconsistent. A diagnosis is related to conflicts such that a set $D \subseteq \mathcal{C}$ is a diagnosis if and only if it has a nonempty intersection with every conflict in a set of conflicts.

## 2.2 Relation Between Local and Global Diagnoses

A minimal local diagnosis is a minimal diagnosis that is determined by the set of conflicts in one agent, while a minimal global diagnosis is determined by all conflicts in all agents. Here, a set of minimal local diagnoses in agent $A$ is denoted by $\mathbb{D}^A$ and a set of minimal global diagnoses is denoted by $\mathcal{D}$.

Due to the distribution of sets of local diagnoses, there is a need to be able to merge these sets of local diagnoses into a set of minimal global diagnoses. For this a merge set is used.

DEFINITION 1 (Merge set): *A* merge set *for a collection of sets of sets* $C$ *is a set* $H \subseteq \bigcup_{S \in C, s \in S} s$, *such that for all* $S \in C$, $s \in S$ *exists such that* $s \subseteq H$.

Similarly to minimal diagnoses, a minimal merge set $H$ is minimal if there is no proper subset $H' \subset H$ where $H'$ is a merge set.

EXAMPLE 2: Consider the collection of sets

$$C = \left\{ \{\{A, B\}, \{C\}\}, \{\{B\}, \{D\}, \{A, C\}\} \right\}.$$

A minimal merge set for this collection is a subset of the set $\{A, B, C, D\}$. One such set is $H = \{A, B\}$, and the set

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

is the complete set of minimal merge sets. ◇

It is straightforward to adapt algorithms used for computing minimal hitting sets, e.g. (Reiter, 1987), such that they can be used to compute minimal merge sets. Later, it will be seen that a merge set operation is useful when calculating the global diagnoses with minimal cardinality. As a partial result, the merge can be used to calculate the minimal global diagnoses from sets of minimal local diagnoses, i.e. $\mathcal{D} = \texttt{MinimalMergeSets}(\{\mathbb{D}^A\}_{A \in \mathcal{A}})$.

## 2.3 Global Diagnoses Represented as Module Diagnoses

If a system can be partitioned into two or more sub-systems that do not share components with each other, then there will be of no practical advantage to merge the minimal diagnoses for all such sub-system even though this would give the minimal global diagnoses. From a technician's point of view, the disjoint sets of minimal diagnoses will in themselves be more easily understandable, since they relate to different independent sub-systems.

This idea can be further exploited by partitioning the set of agents into sub-systems that are independent considering the minimal local diagnoses, even though they might share components that are *not* included in the minimal diagnoses. Each such partition of agents is here denoted a module.

DEFINITION 2 (Module): *Let $\mathbb{D}^{\mathcal{A}}$ be a set of local diagnoses in agent $\mathcal{A}$. A set $\bar{\mathcal{A}} \subseteq \mathcal{A}$ is a* module *if the intersection* $(\cup_{A \in \bar{\mathcal{A}}} \cup_{D \in \mathbb{D}^A} D) \cap (\cup_{A \notin \bar{\mathcal{A}}} \cup_{D \in \mathbb{D}^A} D) = \emptyset$.

The modules are defined with respect to some sets of local diagnoses. Therefore, the modules will be different when different sets of local diagnoses are considered, such as the set of minimal local diagnoses, and the set of local diagnoses with minimal cardinality. A minimal module diagnosis is a minimal diagnosis determined by all conflicts in all agents in one module.

EXAMPLE 3: Figure 3 shows an example of a system that includes five agents. A component is connected with a line to an agent if it is included in the minimal local diagnoses in the agent. It can be seen that the agents can be partitioned into two modules when the sets of minimal local diagnoses are considered. ◇

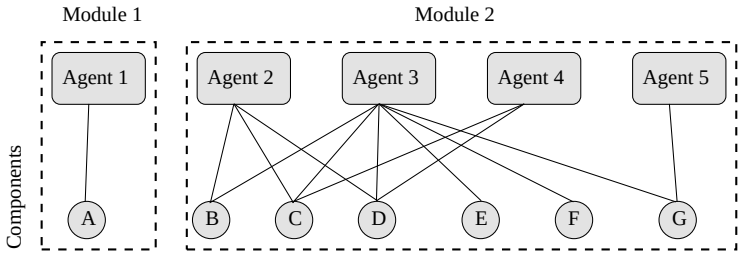



FIGURE 3: The agents have calculated sets of minimal local diagnoses that includes the components connected with lines. The agents have been partitioned into two modules.

The sets of minimal module diagnoses can be computed either from the minimal conflicts, as indicated by the definition, or with the following proposition.

PROPOSITION 1: *Let $\bar{\mathcal{A}}$ be a module, and $\mathbb{D}^{\mathcal{A}}$ be a set of minimal local diagnoses for agent $\mathcal{A}$ determined by the set of conflicts $\Pi^{\mathcal{A}}$, then the set of minimal module diagnoses $\mathcal{D}^{\bar{\mathcal{A}}}$ is the set*

$$\mathcal{D}^{\bar{\mathcal{A}}} = \texttt{MinimalMergeSets}(\{\mathbb{D}^{\mathcal{A}}\}_{\mathcal{A} \in \bar{\mathcal{A}}}).$$

The minimal global diagnoses can be calculated from all minimal module diagnoses, $\mathcal{D} = \texttt{MinimalMergeSets}(\{\mathcal{D}^{\bar{\mathcal{A}}}\}_{\bar{\mathcal{A}}})$, which follows from Proposition 1. The following example is used to illustrate the relation between minimal local diagnoses, minimal module diagnoses, and minimal global diagnoses.

EXAMPLE 4: If a system consists of three agents that have calculated the sets of minimal local diagnoses $\mathbb{D}^{\mathcal{A}_1} = \{\{A, B\}, \{C\}\}$, $\mathbb{D}^{\mathcal{A}_2} = \{\{B, E\},$

$\{C\}\}$, and $\mathbb{D}^{A_3} = \{\{F\}\}$, then the set of agents is partitioned, considering the minimal local diagnoses, into the modules $\bar{\mathcal{A}}_1 = \{A_1, A_2\}$ and $\bar{\mathcal{A}}_2 = \{A_3\}$. This particular partition is a consequence of the fact that component F is included in a set in $\mathbb{D}^{A_3}$ but neither in a set in $\mathbb{D}^{A_1}$ nor in a set in $\mathbb{D}^{A_2}$. The sets of minimal module diagnoses are $\mathcal{D}^{\bar{\mathcal{A}}_1} = \{\{C\}, \{A, B, E\}\}$ and $\mathcal{D}^{\bar{\mathcal{A}}_2} = \{\{F\}\}$.

A merge of the minimal module diagnoses results in the set

$$\texttt{MinimalMergeSets}(\{\mathcal{D}^{\bar{\mathcal{A}}_1}, \mathcal{D}^{\bar{\mathcal{A}}_2}\}) = \{\{C, F\}, \{A, B, E, F\}\}$$

and this is the set of minimal global diagnoses. ◇

## 2.4 *Minimal Cardinality Local, Global, and Module Diagnoses*

Since the problem of computing minimal diagnoses is NP-complete (Friedrich et al., 1990), it is sometimes computationally intractable to calculate the complete set of minimal diagnoses. To reduce the computational cost, the search can be focused on the diagnoses with minimal cardinality, as described in for example (Tuhrim et al., 1991).

The set of minimal cardinality local diagnoses for agent A is denoted by $\mathbb{D}^{A}_{mc}$, the set of minimal cardinality global diagnoses is denoted by $\mathcal{D}_{mc}$, and the set of *minimal cardinality module diagnoses* (MCMD) is denoted by $\mathcal{D}^{\bar{\mathcal{A}}}_{mc}$.

A merge of all minimal local diagnoses does form the set of minimal global diagnoses. If it would be possible to merge the minimal cardinality local diagnoses to gain the minimal cardinality global diagnoses, then a direct approach to calculate the minimal cardinality global diagnoses would be available. Unfortunately, a merge of the minimal cardinality local diagnoses does *not* result in the minimal cardinality global diagnoses, i.e. $\mathcal{D}_{mc} \neq \texttt{MinimalMergeSets}(\{\mathbb{D}^{A}_{mc}\}_{A \in \mathcal{A}})$. They are in general not even a subset of the merged diagnoses. The inequality is exemplified in the following example.

EXAMPLE 5: Consider a system consisting of two agents with the sets of minimal local diagnoses $\mathbb{D}^{A_1} = \{\{C\}, \{A, B\}\}$ and $\mathbb{D}^{A_1} = \{\{E\}, \{A, B\}\}$. The sets of minimal cardinality local diagnoses are $\mathbb{D}^{A_1}_{mc} = \{\{C\}\}$ and $\mathbb{D}^{A_2}_{mc} = \{\{E\}\}$. A merge of these sets of diagnoses give

$$\texttt{MinimalMergeSets}(\{\mathbb{D}^{A_1}_{mc}, \mathbb{D}^{A_2}_{mc}\}) = \{\{C, E\}\}$$

while the set of minimal cardinality global diagnoses is

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

The merge of the minimal cardinality local diagnoses did not result in the set of minimal cardinality global diagnoses. ◇

A merge of the sets of MCMD does however give the minimal cardinality global diagnoses.

PROPOSITION 2: *Let* $\mathcal{D}_{mc}^{\bar{\mathcal{A}}_1}, \ldots, \mathcal{D}_{mc}^{\bar{\mathcal{A}}_n}$ *be the sets of* MCMD, *then the set of minimal cardinality global diagnoses is*

$$\mathcal{D}_{mc} = \texttt{MinimalMergeSets}(\{\mathcal{D}_{mc}^{\bar{\mathcal{A}}_i}\}_{i=1}^{n}).$$

Proposition 2 shows that the definition of a module is sound. However, the actual merge does not have to be performed according to the argumentation in Section 2.3.

# 3 COMPUTING THE MINIMAL CARDINALITY MODULE DIAGNOSES

From the discussions above, it can be concluded that there is an advantage in calculating the set of MCMD compared to the minimal cardinality global diagnoses. If the minimal module diagnoses were available, then the MCMD could directly be identified. On the other hand, if they were not available, then it would not be an efficient approach to first calculate the minimal module diagnoses and then identify the set of MCMD, since it is often more, or even much more, expensive to calculate the minimal module diagnoses than the MCMD. In this paper, the case is studied where neither the minimal module diagnoses nor the minimal global diagnoses in themselves are wanted.

In a distributed environment, the minimal module diagnoses could be calculated by first finding a partition of the agents into modules. After this, the minimal local diagnoses in the agents in each module is merged and this results in the set of minimal module diagnoses.

The set of MCMD can be calculated using a similar method. First, find a lower limit on the cardinality of each element in the set of MCMD and then merge those of the minimal local diagnoses that are smaller or equal to the lower limit, hoping that the elements in the set of MCMD actually have cardinality equal to the lower limit. If no set of MCMD is found, then the lower limit is increased and the merge is started again from the first agent. After some iterations, the lower limit is the cardinality of the elements in the MCMD, and the set of MCMD has then been found. The following example will be used to exemplify the method outlined above.

EXAMPLE 6: Consider a module with three agents including the following sets of minimal local diagnoses

$$\mathbb{D}^{\mathcal{A}_1} = \{\{A, B\}, \{C\}\} \qquad \mathbb{D}^{\mathcal{A}_2} = \{\{A\}, \{B, C, D\}\} \qquad \mathbb{D}^{\mathcal{A}_3} = \{\{D\}\}.$$

If all the local diagnoses with cardinality one were merged and this resulted in a subset of global diagnoses with cardinality one, then this subset would be the set of MCMD. Therefore, first merge the local diagnoses with cardinality one. Agent $A_1$ includes the local diagnosis $\{C\}$ that is transmitted to agent $A_2$ where it is merged with the local diagnosis with cardinality one, resulting in the set $\{A, C\}$. From this it is concluded that the cardinality of each element in the set of MCMD must be two or greater, the search is therefore restarted but with the new lower limit set to two.

Both diagnoses in $A_1$ are transmitted to $A_2$ where they are merged with the diagnosis $\{A\}$ resulting in the set $\{\{A, B\}, \{A, C\}\}$. This set is transmitted to agent $A_3$ where it is merged with $\{\{D\}\}$ resulting in the set $\{\{A, B, D\}, \{A, C, D\}\}$. Since the cardinality of the diagnoses is three, once again, the lower limit has to be raised to three and the search has to be started all over again from agent $A_1$. The last iteration results in the set $\{\{A, B, D\}, \{A, C, D\}, \{B, C, D\}\}$ and this is the set of MCMD. ◇

Even though this simple algorithm does calculate the set of MCMD, the computational cost can be reduced by partitioning the agents into smaller modules, sorting the agents into an appropriate order, and to keep track of which local diagnoses that have been merged in previous iterations. The algorithm designed in this paper uses these improvements.

## 3.1 *Algorithm for Calculating the Set of* MCMD

Algorithm 1 can be used to calculate the set of MCMD and consists of the same three main parts as the simple approach described in the beginning of Section 3. Firstly, algorithm `CalculateModules` is used to partition the set of agents into modules. Algorithm `CalculateOrder` is then used to sort the agents in each module into a merge order R, where $(\cdot)$ is an ordered set. Finally, the minimal local diagnoses are, with algorithm `UpdateAgent`, iteratively merged into sets of MCMD. The correctness of the algorithms is proven in (Biteus, 2005).

Algorithm 1 is designed such that the evaluations of the computationally and memory expensive `UpdateAgent` can be distributed to the corresponding agents. The not so computationally and memory expensive algorithms `CalculateModules` and `CalculateOrder` are evaluated in some coordinating agent.

## 3.2 *Outline of the Algorithm*

Consider Algorithm 1 that ensures the sets of MCMD. First, the modules are calculated and a merge order R is calculated for each module. The algorithm starts with a lower limit $\mathcal{L}$ on the cardinality of each

**Algorithm 1** Minimal cardinality module diagnoses

**Input:** All minimal local diagnoses $\mathbb{D}^{\mathcal{A}}$ for all agents $\mathcal{A}$.
**Output:** All sets of MCMD, where the set of MCMD in module $\bar{\mathcal{A}}$ is the $\mathcal{D}^{\bar{\mathcal{A}}}_{mc}$.

1: $\mathbb{A} :=$ `CalculateModules`$(\mathcal{A})$ [*Set of set of agents.*]
2: **for each** $\bar{\mathcal{A}} \in \mathbb{A}$ **do**
3: $R :=$ `CalculateOrder`$(\bar{\mathcal{A}})$ [*Merge order* $R = (R_1, \ldots, R_n)$.]
4: $\mathcal{L} :=$ a lower bound on the cardinality of each element in the set of MCMD for the module R.
5: $k := 0$, $R_0 := \emptyset$, $N := \emptyset$
6: **repeat**
7: $k := k + 1$
8: $[\mathcal{L}^{new}, N] :=$ `UpdateAgent`$(R_k, R_{k-1}, \mathcal{L}, N)$
9: **if** $\mathcal{L}^{new} > \mathcal{L}$ **then** $k := 0$, $\mathcal{L} := \mathcal{L}^{new}$ [*Start again from* $R_1$.]
10: **until** $k = n$ [$R_n$ *is the last in* R.]
11: $\mathcal{D}^{\bar{\mathcal{A}}}_{mc} :=$ `MinimalDiagnoses`$(N)$ [*The set of* MCMD *in* $\bar{\mathcal{A}}$.]
12: **end for**

element in the set of MCMD. A first approximation of $\mathcal{L}$ is that each element in the set of MCMD must be at least as large as the largest minimal cardinality local diagnosis, considering all agents in R. After the limit has been calculated, an evaluation of `UpdateAgent` is performed in each agent with the objective to find the set of MCMD from the local diagnoses with cardinality less than or equal $\mathcal{L}$.

Algorithm `UpdateAgent` calculates those diagnoses with cardinality less than or equal to $\mathcal{L}$ that would be formed if the minimal local diagnoses, in agents $R_1$ to $R_k$, which have a cardinality less than or equal to $\mathcal{L}$ were merged. The result is the output N that is an input to `UpdateAgent` for the next agent $R_{k+1}$. In an implementation, the result N should be stored locally in the agent and when needed transmitted to the next agent $R_{k+1}$. If any evaluation of `UpdateAgent` results in the output $N = \emptyset$, i.e. no diagnoses could be found with cardinality less than or equal to $\mathcal{L}$, then a new larger $\mathcal{L}^{new}$ is calculated and a new search begins from the first agent $R_1$. This new $\mathcal{L}$ is chosen such that $N \neq \emptyset$ in the next evaluation of `UpdateAgent` for the agent that calculated $N = \emptyset$. If the last agent $R_n$ calculates a non-empty set N, then this set of diagnoses is the set of MCMD for this module and the algorithm terminates.

## 3.3 *Calculating the Modules –* `CalculateModules`

The smallest possible modules are those that are found when only the minimal cardinality global diagnoses themselves are considered.

However, the computational cost to find the set of smallest modules has to be weighed against the reduction in computational cost when the sets of MCMD later are computed. Here, a quite simple approach with low cost is used.

The input to Algorithm `CalculateModules` is the set of agents $\mathcal{A}$, for which the set of components $X^{\mathcal{A}} = \bigcup_{\pi \in \Pi^{\mathcal{A}}} \pi$ has been computed, where $\Pi^{\mathcal{A}}$ is the set of conflicts in agent $\mathcal{A}$. The set $X^{\mathcal{A}} = \cup_{D \in \mathbb{D}^{\mathcal{A}}} D$, since the set of minimal diagnoses are exactly characterized by the set of minimal conflicts, and Definition 2 can therefore be directly translated into an algorithm that partitions the set $\mathcal{A}$ into modules.

## 3.4 *Calculating the Merge Order –* `CalculateOrder`

By changing the ordering of the agents, the computational cost of Algorithm 1 might change dramatically. The cost to calculate the ordering must however once again be weighted against the reduction of computational cost in the rest of Algorithm 1.

Simulation experiments have shown that the computational cost for the rest of Algorithm 1 varies substantially for different orderings of $R$, and unfortunately, it varies also substantially for different local diagnoses for the same type of system. Simulations have shown that an ordering of the agents such that the agent with the smallest number of minimal cardinality local diagnosis is first, and then the rest of the agents follow in ascending number, results in a low average value of the computational cost. The input to `CalculateOrder` is a module found with `CalculateModules`. The output is an ordered set $R \subseteq \mathcal{A}$, where each $R_i \in R$ points at some agent.

## 3.5 *Calculation the Set of* MCMD *–* `UpdateAgent`

Function `UpdateAgent` is shown in Algorithm 2. When the main algorithm evaluates `UpdateAgent` with input $(R_k, R_{k-1}, \mathcal{L})$ this function call is sent to agent $R_k$. The agent that receives this function call should calculate the diagnoses with cardinality less than or equal to $\mathcal{L}$ in the set resulting from the merge of the local diagnoses with cardinality less than or equal to $\mathcal{L}$ in agents $R_1$ to $R_k$. Since agent $R_{k-1}$ has already calculated a set of such diagnoses but for agents $R_1$ to $R_{k-1}$, the desired diagnoses can be calculated from the result in $R_{k-1}$ and the local diagnoses with cardinality less than or equal $\mathcal{L}$ in $R_k$. Variable $l$ is the number of times that the agent has been called with `UpdateAgent`$(\cdot)$. The result of the algorithm (variable $N$) is saved for use by the next agent. If $R_k$ is the last agent and $N \neq \emptyset$ then $N$ is the set of MCMD.

EXAMPLE 7: Consider a module where two agents have calculated the

**Algorithm 2** `UpdateAgent`

**Input:** $R_k$, $R_{k-1}$, $\mathcal{L}$, and $N$. Require $\mathbb{D}^{R_k}$ in agent $R_k$. If $l = 1$, i.e. the first evaluation of the algorithm in $R_k$, then $\mathbb{D}_{previous} := \emptyset$, and $\mathbb{D}_{store} := \emptyset$.

**Output:** Lower limit $\mathcal{L}$ and $N$, which is the subset of merged diagnoses for agents $\{R_1, \ldots, R_k\}$ with cardinality $\leqslant \mathcal{L}$.

1: $\mathbb{D}_{\mathcal{L}} := \{D \in \mathbb{D} : |D| \leqslant \mathcal{L}\}$
2: $\mathbb{D} := \mathbb{D} \backslash \mathbb{D}_{\mathcal{L}}$
3: **if** $k = 1$ **then** [*The first agent in* R.]
4: $N := \mathbb{D}_{\mathcal{L}}$ [*New diagnoses from* $R_k$.]
5: **else if** $k > 1$ **then**
6: $T_l := N$ in $R_{k-1}$ [*New diagnoses from* $R_{k-1}$.]
7: $M_1 := \texttt{MinimalMergeSets}(\{\mathbb{D}_{\mathcal{L}}, \bigcup_{j=1}^{l} T_j\})$
8: $M_2 := \texttt{MinimalMergeSets}(\{\mathbb{D}_{previous}, T_l\})$
9: $\mathbb{D}_{new} := M_1 \cup M_2 \cup \mathbb{D}_{store}$
10: $N := \{D \in \mathbb{D}_{new} : |D| \leqslant \mathcal{L}\}$ [*New diagnoses from* $R_k$.]
11: $\mathbb{D}_{store} := \mathbb{D}_{new} \backslash N$
12: **if** $N = \emptyset$ and $N$ has previously been non-empty **then**
13: $\mathcal{L} := \min(\min_{D \in \mathbb{D}_{store}} |D|, \min_{D \in \mathbb{D}} |D|)$ [*New lower limit.*]
14: **end if**
15: $\mathbb{D}_{previous} := \mathbb{D}_{previous} \cup \mathbb{D}_{\mathcal{L}}$
16: **end if**

sets of minimal local diagnoses

$$\mathbb{D}^{A_1} = \{\{C\}, \{B, D\}, \{B, E, F\}\} \qquad \mathbb{D}^{A_2} = \{\{B\}, \{C, E\}, \{D, E, F\}\},$$

which should be used to calculate the set of MCMD $\{\{B, C\}, \{B, D\}, \{C, E\}\}$. The first value of the lower limit $\mathcal{L}$ is 1, see Section 3.2.

$A_1$ finds the local diagnoses with cardinality less than or equal to $\mathcal{L}$, i.e. diagnosis $\{C\}$, which is transmitted to $A_2$. Agent $A_2$ merges the received diagnosis with its own local diagnoses with cardinality less than or equal to 1, resulting in the set $M_1 = \{\{B, C\}\}$ and this is one of the MCMD.

More MCMD might exist since neither the local diagnosis $\{B, D\}$ nor $\{C, E\}$ have been considered to be part of a minimal cardinality module diagnosis. Agent $A_1$ therefore transmits the diagnosis $\{B, D\}$ to $A_2$. The second agent now merges its own local diagnoses that have *not* previously been considered and have a cardinality less than or equal 2 with all diagnoses received from the first agent, resulting in the set $M_1 = \{\{C, E\}\}$. It also merges the newly received diagnosis with its own local diagnoses that have been considered in the previous iteration, resulting in the diagnoses $M_2 = \{\{B, D\}\}$. To conclude, the set of MCMD is $\{\{B, C\}, \{C, E\}, \{B, D\}\}$. ◇

# 4 EVALUATION OF THE ALGORITHM

To increase the understandability and to point out the advantages of Algorithm 1, it will here be compared to a centralized algorithm that is based on the often used method for computing the minimal diagnoses given in (Kleer and Williams, 1987).

## 4.1 *The Test Suite Used in the Evaluation*

To evaluate the designed algorithm, an application including a distributed system consisting of multiple agents is needed. Unfortunately, only two of the agents in the Scania application described in Section 2 are available and this is not sufficiently many agents since the gains of the designed algorithm is primarily seen for larger systems. Therefore, the algorithm will be evaluated for a suite of test cases whose construction is inspired by diagnostic systems used in automotive applications and notably the diagnostic system described in Section 2. Each test case will be exactly defined by a set of parameters that describes the characteristics of the diagnostic system. Since each test case is defined by a set of parameters, it is possible to evaluate how the complexity and the number and size of the minimal condensed diagnoses scale when the different parameters are changed.

Each test case consists of $n$ agents with $n_t$ diagnostic tests that have responded in each agent, thereby creating $n_t$ conflicts in each agent. The tests that have not been evaluated or have not responded is not included in the definition of the test case. Each test monitors $n_{c/t}$ components[2] with an overlap of $n_{overlap}$ components between the tests. The agents are partitioned into sets of $n_{a/m}$ agents[3] such that there is a connection of $n_{con}$ components between two sequential agents within each set and none between the sets. Each set of connected agents will by this construction be a module. With these parameters, each test case is exactly defined. In the studied automotive application described in the beginning of Section 2, $n_{con}$ is commonly low compared to the number of supervised components, while $n_{overlap}$ might vary from 0 to $n_{c/t} - 1$.

Table 1 shows the isolation structure for the test case where $n_t = 3$, $n_{c/t} = 3$, $n_{overlap} = 1$, $n_{con} = 1$, and $n_{a/m} > 1$. This test case has approximately one and a half faulty components per agent, which can be compared to the automotive application that typically could have between zero and two faulty components in each agent.

[2] $c/t$ = components per test.
[3] $a/m$ = agents per module.

TABLE 1: Isolation structure for a test case that includes 3 diagnostic tests per agent.

| | | Component | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Agent | Test | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $c_7$ | $c_8$ | $c_9$ | $c_{10}$ | ... |
| Agent $A_1$ | $T_{11}$ | × | × | × | | × (overlap) | | | | | | |
| | $T_{12}$ | | | × | × | × | | | | | | |
| | $T_{13}$ | | | × (overlap) | | × | × | × | | | | |
| Agent $A_2$ | $T_{21}$ | | | | | | | × | × | × | | |
| ⋮ | ⋮ | | | | | | | × (connection) | | ⋮ | ⋮ | ⋮ |

## 4.2 *The Centralized Algorithm*

The centralized algorithm is based on the method given in (Kleer and Williams, 1987). In the method, the set of diagnoses is initialized to the empty diagnosis. If a diagnosis has an empty intersection with a conflict, then the diagnosis is extended to several new diagnoses where each new diagnosis includes the old diagnosis and one component from the conflict. The non-minimal extended diagnoses are then removed. The diagnoses are extended until all conflicts have been considered and the result is the set of minimal diagnoses.

In the centralized algorithm, all conflicts from all agents are transmitted to a central agent. The central agent then partitions the conflicts into modules using the same method as used by Algorithm 1. The method in (Kleer and Williams, 1987) is then used to compute the sets of MCMD for each module. However, since only the set of MCMD is wanted, the method is modified such that, after each extension, the diagnoses with a cardinality higher than the cardinality of the MCMD are removed. Simulations indicate that it is most efficient to only remove the non-minimal diagnoses after the last conflict has been considered, therefore this approach is chosen.

## 4.3 *The Number of Needed Operations and Transmissions*

To compare the two algorithms, the number of non-trivial transmissions and operations are computed. For each diagnosis or conflict that is transmitted on the network, the number of transmissions is assumed to be equal to the cardinality of the diagnosis or conflict. The operations that are considered non-trivial are described below.

When partitioning the agents into modules, one agent is first assigned to one module, and to check if the agent shares components with any of the other agents requires one operation per agent not assigned to any module. For the centralized algorithm, one operation

is needed to check if each diagnosis has an empty intersection with a conflict, and one operation is needed to extend the diagnosis. For Algorithm 1, the computation of the local diagnoses, and row 7 and 8 in Algorithm 2 contribute with non-trivial operations. The minimal local diagnoses are computed with the unmodified method in (Kleer and Williams, 1987) and the computation contributes with the operations described above for the centralized algorithm, plus one operation for each extended diagnosis that is checked for non-minimality considering one non-extended diagnosis. Row 7 and 8 have been implemented as a straightforward cross product of the input sets, and therefore contribute with $|\mathbb{D}_{\mathcal{L}}| \cdot |\bigcup_{j=1}^{l} T_j|$ plus $|\mathbb{D}_{previous}| \cdot |T_l|$ operations.

## 4.4 *Comparing the Algorithms*

The complexity of the algorithms can for some test cases be stated analytically. One such test case is the one with parameter values $n_{con} = 0$ and $n_{overlap} = 0$. For such a test case, each agent is a module and the centralized algorithm requires a maximum of

$$\frac{n(n-1)}{2} + 2\frac{n_{c/t}^{n_t} - 1}{n_{c/t} - 1} \cdot n$$

operations in one agent, while Algorithm 1 only requires

$$\frac{n(n-1)}{2} + 2\frac{n_{c/t}^{n_t} - 1}{n_{c/t} - 1}$$

operations. The first part is from the module partitioning and the second from the computation of the sets of MCMD.

For this test suite, Algorithm 1 requires fewer or the same number of operations as the centralized algorithm. For other test cases, if the number of operations that arise from the merge in Algorithm 2, i.e. row 7 and 8, is not sufficiently distributed between the different agents, then Algorithm 1 might be less efficient than the centralized algorithm.

### Evaluation for a test suite of systems

To compare the algorithms for other test cases, the number of operations has been counted during execution. Consider the test suite where $n_t = 3$, $n_{c/t} = 3$, $n_{overlap} = 1$, $n_{a/m} = 4$, $n_{con} = 1$, and the number of agents $n$ is varied between one and twelve. The sets of MCMD have been calculated using the two algorithms and the number of needed operations and transmissions have been counted. Figure 4 shows the maximum number of operations in any agent and the sum

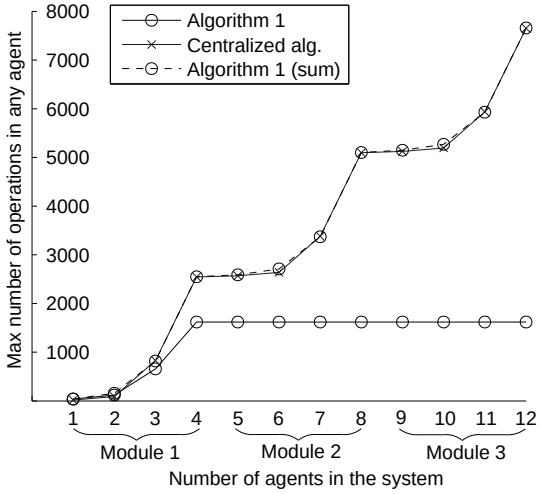



FIGURE 4: Max number of operations in any agent.

of all operations in all agents for Algorithm 1. Due to the number of agents per module $n_{a/m} = 4$, the set of agents is partitioned into modules such that agent one to four is included in the first module, five to eight in the second, etc. It can be seen that the maximum load increases exponentially when the size of the systems are increased from one to four agents since the first module includes up to $n_{a/m} = 4$ agents. For systems with five or more agents, the maximum load for Algorithm 1 reaches a constant value due to the distribution of processor load onto the different agents. The maximum load for the centralized algorithm increases linearly with the number of modules since all sets of minimal cardinality module diagnoses are computed in the same agent. The linearity can for example be seen by considering the number of operations for four, eight, and twelve agents.

If the module partitioning had not been used, then the number of operations will continue to grow exponentially for systems with 5 or more agents. The module partitioning reduces the complexity from exponentially to linearly increasing. The difference in complexity growth gives a reduction of 40 % in the maximum load on any agent when the system consists of four agents, and it increases to 70 % when the system consists of eight agents. Evaluations have shown that if the centralized algorithm computes the minimal cardinality global diagnoses instead of the minimal cardinality module diagnoses, then the reduction in computational load would be much higher, for example is a reduction of over 99.9 % achieved for the system with eight agents. The sum of operations shows that there is no major overhead for Algorithm 1 compared to the centralized algorithm.

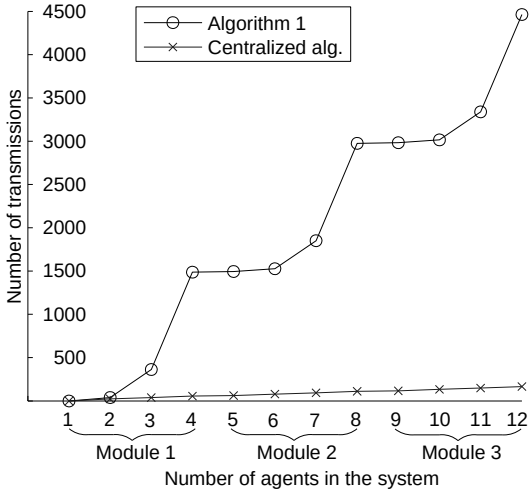



FIGURE 5: Number of transmissions on the network.

The number of operations needed for computing the module partitioning increases linearly with the number of agents in the system and is included in the number of operations. However, evaluations have shown that for this tests suite these operations do not start to affect the maximum number of operations in Algorithm 1 until the 109:th agent.

For the test suite, the number of transmissions increases linearly with the number of modules for Algorithm 1 and with the number of agents for the centralized algorithm, see Figure 5.

### Conclusions from the evaluation

The important conclusions from this test suite are that, Algorithm 1 is more efficient than the centralized algorithm for the test cases with three agents or more and that both the gain in operations and the cost in transmissions for Algorithm 1 increase linearly with the number of modules.

Is Algorithm 1 always more efficient than the centralized algorithm for systems with three agents or more? To answer this question, a test suite is constructed inspired by the automotive application. The variables $n_t$ and $n_{c/t}$ are varied between one and three, $n_{con}$ is varied between 0 and $n_{c/t} - 1$, and $n$ is varied between one and twelve. Similar to the test suite studied above, the designed algorithm is also for this test suite on average more efficient than the centralized algorithm for systems with three agents or more.

### 4.5 *Efficiency of The Algorithm*

The reduction of maximum processor load on any agent when using Algorithm 1 comes from the fact that each agent computes its minimal local diagnoses, which requires fewer operations than the computation of the minimal cardinality global diagnoses using the centralized algorithm, and that the evaluations of Algorithm 2 are distributed to the different agents. The reason why the number of operations needed by Algorithm 2 does not increase above those needed for the centralized algorithm is that, on average, the conflicts in one agent have more components in common between themselves, compared to how many they have in common with the conflicts in the other agents. When this is the case, it is more likely that a local diagnosis with low cardinality is part of one of the MCMD and it is therefore more likely that some of the first iterations in Algorithm 1 results in the MCMD.

An important conclusion from the evaluation is that the partitioning of the system into modules, based on the diagnostic test results, reduces the computational load from exponentially to linearly increasing in the number of modules, independent of the fault isolation algorithm used within each module. The partitioning can be used for all systems where the set of agents can be partitioned into modules with respect to the test results. This is for example the case for a studied automotive vehicle from Scania. In many cases, the distributed system in the Scania vehicle can be partitioned into several modules since the diagnostic tests in one agent mostly supervise components only supervised by that agent. For example, the engine management system (EMS) supervises mostly engine components. However, since tests exist that supervise components also supervised by other tests in other agents, it is not possible to partition the system into modules before the diagnostic test results are known.

## 5 REDUCING THE SIZE OF THE MODULES

In Section 3.3, the set of agents is partitioned into modules such that each minimal conflict in one module has an empty intersection with all other minimal conflicts in the other modules. The same partitioning would be achieved if the set of agents is partitioned with respect of set of minimal diagnoses. However, it is possible to improve the partitioning of the set of agents such that the size of the modules are further reduced. By reducing the size of the modules the complexity when computing the minimal cardinality module diagnoses can be reduced. Consider for example a system with two agents that have the

following sets of minimal local diagnoses.

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

When partitioning the set of agents $\mathcal{A} = \{A_1, A_2\}$ into modules with respect to the minimal diagnoses then both agents will be included in one single module. To compute the set of minimal cardinality module diagnoses given this partitioning, the local diagnosis $\{A\}$ is transmitted to the second agent where the set of minimal cardinality module diagnoses $\mathcal{D}_{mc}^{\{A_1,A_2\}} = \{\{A,B\}\}$ is computed. However, this partitioning can be improved by noticing that the local diagnosis $\{C,D,E\}$ has a cardinality larger than the cardinality of a minimal cardinality global diagnosis and it can therefore not be part of a minimal cardinality module diagnosis. By ignoring this diagnosis, the set of agents can be partitioned into the two modules $\{A_1\}$ and $\{A_2\}$. The set of minimal cardinality module diagnoses are now directly available from the local diagnoses

$$\mathcal{D}_{mc}^{\{A_1\}} = \{\{A\}\} \qquad \mathcal{D}_{mc}^{\{A_2\}} = \{\{B\}\}.$$

As should be, the minimal cardinality module diagnoses are independent of each other. To conclude, the complexity when computing the sets of minimal cardinality module diagnoses can be reduced by improving the partitioning of the set of agents.

In this section, an algorithm that computes such an improved module partitioning will be designed. In Algorithm 1, the improved algorithm can be used instead of function `CalculateModules`$(\cdot)$.

## 5.1 *Algorithm for the Module Partitioning*

The main idea in Algorithm 3 is to find an upper limit $\mathcal{U}$ of the cardinality of the MCMDsand use this to reduce the size of the modules. All minimal local diagnoses with a cardinality higher than this limit can not be part of a MCMD. Such as the diagnosis $\{C,D,E\}$ in the example above.

An upper limit for a set of agents $\bar{\mathcal{A}}$ is

$$\texttt{UpperLimit}(\bar{\mathcal{A}}) := \sum_{A\in\bar{\mathcal{A}}} \min_{D\in\mathbb{D}^A} |D|$$

and this is the sum of the minimal cardinality local diagnoses. In the example above, this upper limit is $2 = |\{A\}| + |\{B\}|$ for the set of all agents. An upper limit that sometimes is even lower can be found by making a pre-merge of all minimal cardinality local diagnoses in the set of agents,

$$\bar{\mathbb{D}} := \texttt{MinimalMergeSets}(\{\mathbb{D}_{A_1}^{mc}, \mathbb{D}_{A_2}^{mc}, \ldots\}).$$

**Algorithm 3** `CalculateModules` version 2.

**Input:** Set of agents $\bar{\mathcal{A}} \subseteq \mathcal{A}$. Require, for each agent, the set of minimal local diagnoses $\mathbb{D}^{\mathcal{A}}$ stored in the agent.

**Output:** A collection of sets of agents $\mathbb{A}$ where each set of agents $\bar{A} \in \mathbb{A}$, $\bar{A} \subseteq \bar{\mathcal{A}}$ is a module.

1: $\mathcal{U} := \texttt{UpperLimit}(\bar{\mathcal{A}})$
2: **for each** $A_i \in \bar{\mathcal{A}}$ **do**
3: $\quad \mathbb{D}^{\mathcal{A}_i} := \{D \in \mathbb{D}^{\mathcal{A}_i} : |D| \leqslant \mathcal{U}\}$
4: $\quad C^{\mathcal{A}_i} := \cup_{D \in \mathbb{D}^{\mathcal{A}_i}} D$ [*All components in the diagnoses*]
5: **end for**
6: $X := \{\bar{A} : (A_i \in \bar{A}) \wedge (A_j \in \bar{\mathcal{A}} \backslash \bar{A}) \wedge (C^{\mathcal{A}_i} \cap C^{\mathcal{A}_j} = \emptyset)\}$ [*Partition.*]
7: **if** $|X| > 1$ **then** [*More than 1 module.*]
8: $\quad \mathbb{A} := \emptyset$
9: $\quad$ **for each** $\bar{A} \in X$ **do**
10: $\quad\quad \mathbb{A} := \mathbb{A} \cup \texttt{CalculateModules}(\bar{A})$ [*Recursive.*]
11: $\quad$ **end for**
12: **else**
13: $\quad \mathbb{A} := X$
14: **end if**

The upper limit is then chosen as the size of a minimal cardinality diagnosis in the set $\bar{\mathbb{D}}$. To reduce the complexity it is possible to only consider the minimal cardinality diagnoses after each merge, which would give

$$\bar{\mathbb{D}} := (\ldots \texttt{MMS}(\{\texttt{MMS}(\{\mathbb{D}^{mc}_{A_1}, \mathbb{D}^{mc}_{A_2}\})^{mc}, \mathbb{D}^{mc}_{A_3}\})^{mc} \ldots)$$

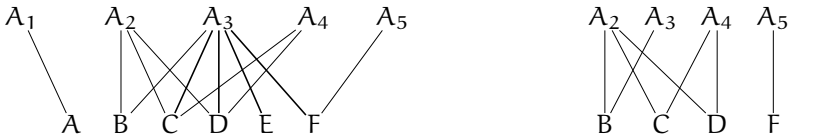
where `MMS` is the function `MinimalMergeSets`. The upper limit is once again chosen as the size of a minimal cardinality diagnosis in the set $\bar{\mathbb{D}}$. This limit requires less computations than the complete premerge but might have a higher value.

For each module, Algorithm 3 is called recursively so that each module is, if it is possible, partitioned into even smaller modules, see row 10.

EXAMPLE 8: Consider a system with five agents including the minimal local diagnoses

$$\mathbb{D}^{\mathcal{A}_1} = \{\{A\}\} \qquad \mathbb{D}^{\mathcal{A}_2} = \{\{B, C\}, \{D\}\}$$
$$\mathbb{D}^{\mathcal{A}_3} = \{\{B\}, \{C, D, E, F\}\} \qquad \mathbb{D}^{\mathcal{A}_4} = \{\{C\}, \{D\}\}$$
$$\mathbb{D}^{\mathcal{A}_5} = \{\{F\}\}$$

Using Algorithm 3 with the second type of upper limit give the premerge $\bar{\mathbb{D}} = \{\{A, D, B, F\}\}$ and the upper limit is therefore $\mathcal{U} = 4$. A




(a) Including all minimal local diagnoses.

(b) Some local diagnoses have been removed.

FIGURE 6: The bipartite graph represent a system with five agents and six components. An agent is connected to a component if the component is included in the agents minimal local diagnoses.

bipartite graph, illustrating the agents and the components included in the minimal local diagnoses, is shown in Figure 6(a). Evaluation of Algorithm 3 give the partitioning $\mathbb{A} = \{\{A_1\}, \{A_2, A_3, A_4, A_5, \}\}$

For the second module, $\bar{A} = \{A_2, A_3, A_4, A_5, \}$, the upper limit is $\mathcal{U} = 3$ and the local diagnosis $\{C, D, E, F\}$ can therefore be removed, since it can not be included in an MCMD. The result is illustrated with the bipartite graph shown in Figure 6(b). The evaluation of Algorithm 3 with $\bar{A} = \{A_2, A_3, A_4, A_5, \}$ as input gives the partitioning $\mathbb{A} = \{\{A_2, A_3, A_4\}, \{A_5, \}\}$. The final result is that the set of agents is partitioned into the three modules $\{A_1\}$, $\{A_2, A_3, A_4\}$, and $\{A_5\}$. ◇

## 5.2 *Evaluation of the Improved Module Partitioning*

Algorithm 3 can in some cases partition the set of agents into smaller modules compared to the function `CalculateModules`$(\cdot)$ used in Algorithm 1. However, evaluations have shown that the mean gain in using the improved algorithm is often lower than the mean cost in evaluating the algorithm. This evaluation has been performed for a test suite that is based on a randomized model, which resembles the distributed systems used in Scania automotive vehicles. In (Biteus et al., 2005), the improved algorithm is used to partition set of agents and even though a small gain is shown, further evaluations has shown that the gain is mostly insignificant or zero.

However, as exemplified in the beginning of this section, systems exist for which the improved partitioning does improve the efficiency when using Algorithm 1. If the minimal cardinality module diagnoses should be computed for such a system then the improved algorithm could be used to reduce the complexity when evaluating Algorithm 1.

# 6 CONCLUSIONS

An algorithm has been designed that uses the sets of minimal local diagnoses in the agents to calculate the module diagnoses with minimal cardinality. The sets of module diagnoses with minimal cardinality are independent of each other and these therefore exactly represent the set of global diagnoses with minimal cardinality. The partitioning of the agents into modules is based on the diagnostic test results and has to be performed after the results are available. The main advantage of the module diagnoses, compared to the global diagnoses, is that the maximum load on any agent is reduced from exponentially to linearly increasing, considering the number of modules in the system. For a diagnostic system inspired by an automotive application that consists of eight agents, the module partitioning gave a reduction of over 99.9 % in maximum processor load on any agent. After the module partitioning, the algorithm computes the set of minimal cardinality module diagnoses in each module by distributing the computationally intensive tasks to the different agents in the module. The distribution further reduces the maximal processor load on any agent from linearly increasing to become constant for systems larger than a certain number of agents. For the system inspired by the automotive application, this constant value is for example reached for systems with four or more agents. The algorithm is designed such that overhead due to the distribution of computations is low and, for example, a reduction of 40 % is achieved within a module consisting of four agents.

# *Paper II*

# DISTRIBUTED DIAGNOSIS USING A CONDENSED REPRESENTATION OF DIAGNOSES WITH APPLICATION TO AN AUTOMOTIVE VEHICLE[1]

**Jonas Biteus[⋆], Erik Frisk[⋆], and Mattias Nyberg[†]**

[⋆] *Dep. of Electrical Engineering, Linköpings universitet,* SE-*581 83 Linköping, Sweden.* {biteus,frisk}@isy.liu.se.
[†] *Power-train division, Scania,* SE-*151 87 Södertälje, Sweden.* mattias.nyberg@scania.com

## ABSTRACT

In fault detection and isolation, diagnostic test results are commonly used to compute a set of diagnoses, where each diagnosis points at a set of components that might behave abnormally. In distributed systems consisting of multiple control units, it is an advantage for both repair and for fault tolerant control to have access to the global diagnoses in each unit since these diagnoses represent all test results in all units. However, the global diagnoses include components from the complete system, and the global diagnoses therefore often include many components that are not affecting the unit and are thereby superfluous. Motivated by this observation, a new novel type of diagnosis is designed, the condensed diagnosis. Each unit has a unique set of condensed diagnoses that only includes the affecting components while still being globally correct. For a studied heavy duty vehicle, the mean number of condensed diagnoses and the maximum processor load on any unit are reduced with up to 90 and 85 % respectively, compared to the global diagnoses.

[1] An earlier and shorter version of this paper has been presented in (Biteus et al., 2006a).

# 1 INTRODUCTION

Fault diagnosis is becoming more common in many industrial applications, and one of the most widespread approaches for diagnosis is the consistency based diagnosis approach developed within the AI field (Kleer and Kurien, 2003; Dressler and Struss, 1996), which has strong relationships with the methods for fault diagnosis used in the engineering disciplines (Cordier et al., 2004; Trave-Massuyes et al., 2006). In this approach, a minimal diagnosis is a minimal set of components whose abnormal behavior is a possible explanation to why the system is faulty. Such minimal diagnoses are commonly used within the AI field for both repair and fault tolerant control. In automotive applications, a diagnosis could for example state that an ambient temperature sensor is behaving abnormally, or that the inlet manifold temperature sensor and the inlet cooler are behaving abnormally.

This paper considers fault diagnosis for distributed systems that consist of a set of agents, where an agent is a more or less independent software entity (Hayes, 1999; Weiss, 1999). In such distributed systems, the global diagnoses are diagnoses for the complete distributed system and can be computed from all diagnostic test results in all agents. Further, the local diagnoses are diagnoses for a single agent and can be computed from the diagnostic test results in that agent. The minimal local diagnoses can for example be used when the agent should be repaired, by guiding the repair technician to the faulty components. However, a drawback with the minimal local diagnoses is that they do not use all diagnostic test results that exist in the complete system. It might for example be the case that one agent has detected, using some diagnostic test, that a component, connected to and used by another agent, is behaving abnormally. The information about this abnormal component would not be available to the second agent if it only had access to its own set of minimal local diagnoses. This is in contrast to the minimal global diagnoses where the information about this abnormal component would be included. It is therefore an advantage if the minimal global diagnoses are available in every agent.

A drawback when using the minimal global diagnoses in an agent is that they include many components that could not affect the behavior of the agent since they are not used in the system controlled by the agent. These unused components make the number of global diagnoses to be unnecessary high and each global diagnosis unnecessary large. When performing for example repair of the agent, all components not used by the agent are can be ignored by the repair technician since could not have caused the abnormal behavior of the agent. In for example a studied automotive application from Scania, the engine control agent does not use the catalytic converter component and it is therefore not interested in the behavior of the catalytic converter.

Therefore, the catalytic converter component could be removed from the global diagnoses if they are used in the engine control agent. Besides increasing the number and the size of the global diagnoses, the inclusion of the unused components in the minimal global diagnoses leads to an unnecessary high use of memory and processing power, and thereby leads to unnecessary costs.

### Condensed diagnoses

Motivated by the observations above, this paper contributes with a novel type of diagnosis, denoted the condensed diagnosis, which is used to represent the set of minimal global diagnoses in each agent. Each agent has a unique set of minimal condensed diagnoses that only includes the global diagnoses including components used by the agent. Further, in these included global diagnoses, all components not used by the agent are removed.

To secure that the condensed diagnoses represent the global diagnoses, a scalar variable is added to each condensed diagnosis whose value equals the number of removed unused components. In for example the automotive application, the condensed diagnoses in the engine control agent does only include components used by the engine, such as an inlet manifold sensor, injection actuators, fuel pipes, etc. Other components are represented by the scalar variable, such as the catalytic converter and the climatic control components.

Due to the removal of the unused components, both the memory needed to store and the processing power needed to compute the sets of minimal condensed diagnoses are reduced compared to when the set of minimal global diagnoses is used. In addition, both the size and the number of minimal condensed diagnoses in each agent are also reduced compared to the minimal global diagnoses. However, in contrast to the minimal local diagnoses that are also smaller than the minimal global diagnoses, the minimal condensed diagnoses still represent the global diagnoses. To compute the condensed diagnoses, this paper contributes with an algorithm that efficiently computes a unique set of minimal condensed diagnoses in each agent. The benefits of the condensed diagnoses are in this paper illustrated for the distributed system in an automotive vehicle, notably a heavy duty truck from Scania.

### Focusing the diagnoses

Above, the set of minimal diagnoses is discussed. However, sometimes, the set of minimal diagnoses are focused on to some smaller set of diagnoses, such as the set of diagnoses with minimal cardinality (Tuhrim et al., 1991), i.e. minimal number of included components.

The reason for this focusing is that it is in some cases computationally intractable to compute the complete set of minimal diagnoses or it is only the diagnoses with minimal cardinality that are used in repair or fault tolerant control. If for example one diagnosis includes one abnormal component, and another includes two abnormal components, then repair is mostly started by first checking the component in the small diagnosis since it is often more likely that one component is abnormal compared to two components.

Since minimal cardinality diagnoses and thereby minimal cardinality global diagnoses are wanted in some applications, the designed algorithm is in this paper extended such that it can be used to efficiently compute a unique set of minimal cardinality condensed diagnoses in each agent. Also this extended algorithm will be applied to the distributed system in the automotive vehicle.

**Distributed systems**

As indicated by the application, our work is inspired by diagnostic systems used in automotive vehicles (Navet et al., 2005; Leen and Heffernan, 2002; Gertler, 1998; Hristu-Varsakelis and Levine, 2005; Struss and Price, 2003). These systems typically consist of precomputed diagnostic tests that are evaluated in the different agents, which in the automotive industry correspond to electronic control units (ECUs).

The automotive distributed systems typically consist of ECUs with both limited processing power and limited RAM memory that motivates the use of the condensed diagnoses. Another important characteristic of these distributed systems are that agents can be disconnected, replaced, or added to the system without notifications. The changes occur for example due to repair and the addition or removal of auxiliary equipments to the vehicle. To gain a scalable (Tanenbaum and Steen, 2002) distributed system that can handle such changes, the use of a dedicated diagnostic agent that computes the sets of minimal condensed diagnoses and then distributes these to the other agents is not desirable. Therefore, the algorithm is designed such that it, for each agent, computes the set of minimal condensed diagnoses in a distributed cooperation between the agents. Using the algorithm, each agent updates its set of minimal local diagnoses such that it becomes a unique set of minimal condensed diagnoses.

## *1.1 Related Work*

In contrast to the distributed algorithm designed in this paper, most research on fault isolation, such as (Reiter, 1987), aim at the centralized diagnosis problem. These centralized methods can also be used for distributed systems by letting a single diagnostic agent collect all

diagnostic test results from all agents and then compute the minimal global diagnoses. However, as noted above, it is not always suitable to use such a dedicated single diagnostic agent. Therefore algorithms exist that perform distributed fault diagnosis by computing the set of minimal global diagnoses in a distributed cooperation between the agents, see for example (Provan, 2002). In (Provan, 2002), the set of minimal global diagnoses is computed for a distributed system using a tree representation of the structure between the agents. The method assumes that each agent states a local diagnosis based on only the internal sensors and component descriptions, i.e. signals from other agents are not used. This is a difference against the method presented in this paper where such an assumption is not made.

**Updating the local fault diagnosis**

Algorithms also exist, such as (Roos et al., 2003), where the agents update the local conflicts such that the set of local diagnoses is globally correct. The local conflicts are updated by updating the knowledge of the behavior of the signals transmitted between the agents, such that the set of minimal local diagnoses in each agent are consistent with the behavior of the set of signals. This method has similarities with the method in this paper that updates the sets of local diagnoses such that they are globally correct.

A difference between the method in this paper and the one in (Roos et al., 2003) is that in the later, the diagnoses are computed based on the model of the system and the observations, while this paper computes the diagnoses based on the diagnostic test results. To compute the diagnoses directly based on the model and the observations as is done in (Roos et al., 2003) is a more general framework. However, in automotive applications, it is commonly the case that tests have been designed and are already implemented in the diagnostic system. This thesis has therefore focused on systems where a set of diagnostic tests is available and fault isolation is performed after some tests have been evaluated. The algorithms designed in this thesis can therefore utilize these tests when performing fault diagnosis instead of performing fault diagnosis directly on the observations.

Due to the more general framework used in (Roos et al., 2003), the algorithm requires an unknown number of iterations before finishing, while the algorithm designed in this paper terminates after one iteration. Another difference is that in (Roos et al., 2003), each resulting diagnosis will only include components and not signals since the signals are replaced with the components on which the signals depend. If a signal depends on two or more components then the replacement makes the number of diagnoses to be increased compared to the method designed in this paper that preserves the signals. Fur-

ther, by preserving the signals in the diagnoses, it is directly known if a signal is behaving abnormally instead of having to know that if a certain component is abnormal then the signal is abnormal. It is often the case that the component might not be used by the agent and it is therefore not necessary to know its status, and it is therefore an advantage to keep the signals when the diagnoses are used in for example fault tolerant control.

Besides consistency based diagnosis, Fault diagnosis based on discrete event systems has also been extensively studied, see e.g. (Lunze and Schröder, 2001; Sampath et al., 1995; Su and Wonham, 2005; Debouk et al., 2000). These methods are based on an automata description that models the discrete transitions between different states. Fault diagnosis of an automata aims to detect and isolate abnormal components by detecting abnormal transitions. This differs from this thesis where a set of diagnostic tests are evaluated that results in a set of conflicts. The diagnostic tests typically supervise the continuous values of signals that are not modeled as discrete transitions.

#### Focusing the diagnoses

In this paper, the set of diagnoses with minimal cardinality is computed. An alternative to compute the set of minimal cardinality diagnoses is to compute the set of most probable diagnoses, which would give better results. In for example (Kleer, 1991), the set of most probable diagnoses is computed using a centralized method. However, it is computationally more difficult to compute this set and it is in many applications difficult, or practically impossible, to get good a-priori probabilities for the failure of the components. The focusing to the most probable diagnoses can therefore be difficult to use.

## 2 DIAGNOSIS IN THE AUTOMOTIVE INDUSTRY

In Figure 1, one configuration of the embedded system used in heavy duty vehicles from Scania is shown. This system includes three separate controller area network (CAN) buses that connect the ECUs to each other. Each of the ECUs is physically connected to some sensors and actuators, and both sensor values and control signals can be shared with the other ECUs over the network. There can be up to about 30 ECUs in the system, depending on the type of truck, and roughly between 4 and 110 components are supervised by each ECU. A small, but not insignificant, number of components are used by several ECUs, which motivates the employment of the condensed diagnoses since these only include the components used by the specific ECU.

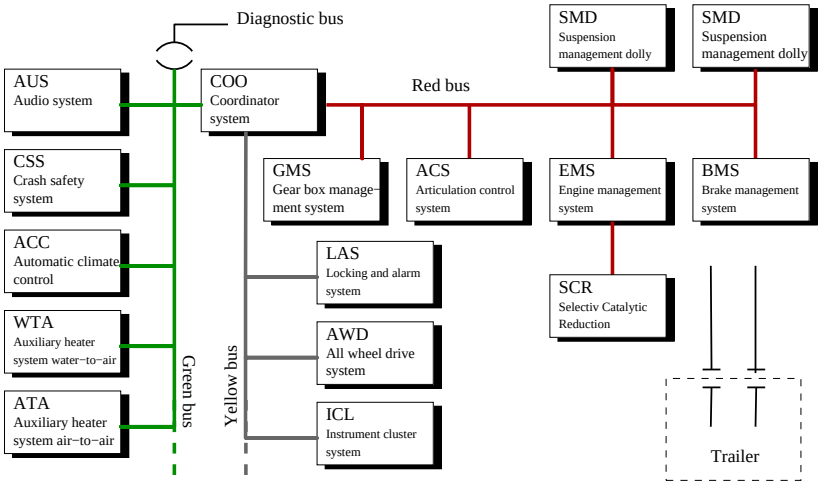



FIGURE 1: The distributed system in current Scania vehicles.

The CPUs in the ECUs have typically a clocking speed of 8 to 64 MHz, and a RAM memory capacity of about 4 to 150 kB. Due to the limited resources, there are in the Scania application not much computational power nor memory capacity available, which motivates the use of the condensed diagnoses since these require low memory capacity and low computational power, compared to when the minimal global diagnoses are used.

The application, for which the algorithm designed in this paper will be applied, consists of the engine management system (EMS), the selective catalytic reduction system (SCR), and the coordinator system (COO), which all can be seen in Figure 1. The EMS controls and supervises the engine, the SCR is an emission control system, while the COO has some coordinating functions.

## 3 CONSISTENCY BASED DIAGNOSIS

A system consists of a set of components $\mathcal{C}$ that should be supervised for abnormal behavior. A component is something that can be diagnosed, such as sensors, actuators, cables, and pipes, and it can be in different behavioral modes. Here, only the abnormal and the not abnormal mode is considered, where the abnormal mode does not have a model. Further, the set notation used in for example GDE (Kleer and Williams, 1987) is used to represent diagnoses, etc.

A diagnosis D is a set of components, such that the abnormal behavior of the components in D, the normal behavior of the remaining

components, the system description, and the observations are consistent. All supersets of a diagnosis $D$ are also diagnoses since only the abnormal mode, without a model, and the normal mode are considered. Further, a diagnosis $D$ is a minimal diagnosis if there is no proper subset $D' \subset D$ where $D'$ is a diagnosis (Kleer et al., 1992).

A conflict is a set of components $\pi \subseteq \mathcal{C}$, such that the normal behavior of the components, the system description, and the observations are inconsistent. As with diagnoses, a conflict $\pi$ is a minimal conflict if there is no proper subset $\pi' \subset \pi$ where $\pi'$ is a conflict. If a diagnostic test responds, i.e. if it detects that any component it supervises is behaving abnormally, then a conflict consisting of the supervised components is generated.

From the definitions follow that a set $D \subseteq \mathcal{C}$ is a diagnosis if and only if it has a nonempty intersection with every conflict. A consequence of this is that the set of minimal diagnoses is the set of minimal hitting sets for the set of minimal conflicts (Reiter, 1987; Kleer et al., 1992), where a hitting set for the set of sets $\Pi$ is a set $D \subseteq \bigcup_{\pi \in \Pi} \pi$ such that $D \cap \pi \neq \emptyset$ for all $\pi \in \Pi$. A minimal hitting set is defined in the same way as minimal diagnoses and minimal conflicts.

EXAMPLE 1: Consider a diagnostic test that supervises two components, denoted component $A$ and $B$. If the diagnostic test has responded then a conflict $\{A, B\}$ would be generated. Similarly, if components $B$ and $C$ were supervised by a test that has responded, then a conflict $\{B, C\}$ would be generated. The set of minimal hitting sets for the set of sets $\{\{A, B\}, \{B, C\}\}$ is the set $\{\{B\}, \{A, C\}\}$ and this is the set of minimal diagnoses. The diagnoses states that either is component $B$ behaving abnormally or both components $A$ and $C$ are behaving abnormally. $\diamond$

# 4 DISTRIBUTED DIAGNOSIS

This section will first give a framework for distributed systems consisting of agents, components, and signals. The framework will then be used when a condensed diagnosis is defined.

## *4.1 Relation Between Local and Global Diagnoses*

A distributed system consists of a set of agents $\mathcal{A} = \{A_1, \ldots, A_n\}$ where each agent includes a diagnostic system. In such distributed systems, a local diagnosis is a diagnosis that is determined by the set of conflicts in one agent, while a global diagnosis is determined by all conflicts in all agents. The conflicts are typically generated from responded diagnostic tests. Here, the set of minimal local diagnoses in

agent $A_i$ is denoted by $\mathbb{D}^{\mathcal{A}_i}$ and the set of minimal global diagnoses is denoted by $\mathcal{D}$.

The set of global diagnoses can be computed from the sets of minimal conflicts in all agents. However, a merge of all sets of minimal local diagnoses also results in the set of minimal global diagnoses. If for example the two sets of minimal local diagnoses $\{\{A\},\{B\}\}$ and $\{\{C\}\}$ have been computed, then a cross product of these sets gives the set $\{\{A, C\},\{B, C\}\}$, which is the set of minimal global diagnoses. Even though the cross product is used in the example to merge the minimal local diagnoses such that the minimal global diagnoses are computed, this can not be done in the general case. If for example the sets of minimal local diagnoses $\{\{A\},\{B\}\}$ and $\{\{A\}\}$ are merged with a cross product, then the result is the set $\{\{A\},\{A, B\}\}$ where $\{A, B\}$ is not a minimal global diagnosis. Therefore, to gain the set of minimal global diagnoses, the non-minimal global diagnoses have to be removed when the minimal local diagnoses are merged.

This idea of computing the minimal global diagnoses from the sets of minimal local diagnoses will be used in the algorithm that computes the set of minimal condensed diagnoses in each agent.

## *4.2 Signals and Components in Distributed Systems*

A condensed diagnosis in one agent should only include the components that the agent uses. To be able to decide which components that an agent use, the components are here partitioned into private components $\mathcal{P} \subseteq \mathcal{C}$ and common components $\mathcal{G} \subseteq \mathcal{C}$. A private component is only used by one agent, while a common component is used by two or more agents. The set of private components is therefore further partitioned into different sets belonging to different agents, where the set $\mathcal{P}^{\mathcal{A}_i} \subseteq \mathcal{P}$ is used by agent $A_i$. For the automotive application, the private components in the EMS are for example sensors measuring inlet manifold temperature and pressure, diesel injection actuators, and fuel pipes. The common components are for example the battery voltage sensor and the network cables. Other partitionings of the components could be considered, for example to partition the common components into subsets of components used by different subsets of agents. However, both the definition of condensed diagnosis and the algorithm for the computation of the minimal condensed diagnoses would with this partitioning become more cumbersome. Therefore, to keep the presentation clear, the simpler partitioning will be used.

An agent in a distributed system could, in addition to the components, also use signals available over the network. A signal is typically a value from a sensor, to an actuator, or some computed value. Here $\mathcal{S}$ is the set of signals, $\mathcal{IN}^{\mathcal{A}_i} \subseteq \mathcal{S}$ is the subset of signals that are used by agent $A_i$ and $\mathcal{OUT}^{\mathcal{A}_i} \subseteq \mathcal{S}$ is the subset of signals that are outputs from

agent $A_i$. In the automotive application, one signal is for example the value from the sensor measuring the inlet manifold temperature, and another signal is the amount of catalytic reduction fluid that has been injected into the catalytic converter.

## 4.3 Signals Depending on Components

The normal behavior of a signal $s$ depends on the normal behavior of all the components in a set $\mathtt{Dep}(s) \subseteq \mathcal{C}$. If any of the components in the set $\mathtt{Dep}(s)$ behaves abnormally then $s$ is assumed to also behave abnormally. As mentioned above, in the automotive application, one signal is the value from the inlet manifold temperature sensor and another signal is the amount of injected catalytic reduction fluid. The temperature signal only depends on one the manifold temperature sensor component, while the fluid amount signal depends on several components, such as the injection actuator and a temperature sensor in the fluid tank.

To make the algorithm for the computation of the minimal condensed diagnoses more readably, some special properties of the dependencies will be assumed.

ASSUMPTION 1: *a) The dependency for a signal $s$ that is an output from agent $A_i$ is limited such that $\mathtt{Dep}(s) \subseteq \mathcal{P}^{A_i}$. b) The dependencies for two different signals are disjoint.*

These assumptions make the algorithm more readable, but as shown in (Biteus et al., 2006a), in Section 9.2, and in Section 9.2, it is possible to compute the minimal condensed diagnoses even if Assumption 1 is not fulfilled. Besides making the algorithms more readable, it is also the case that Assumption 1 is often fulfilled in industrial applications by construction. This is the case for the automotive application studied later in this paper. For example is b) often fulfilled since if this is not the case then the signals can not be used for supervision independently of each other.

Since the diagnostic system in an agent could supervise both components and signals, a diagnosis $D \subseteq \mathcal{C}$, as defined in Section 3, has to be extended to the set of components and signals. However, a diagnosis including signals can always be propagated to the set of components by replacing the signals with their dependencies. Due to this propagation, the word diagnosis will here be used both for a true diagnosis and a diagnosis including both components and signals.

## 4.4 Condensed Diagnoses Representing Global Diagnoses

In this section, a minimal condensed diagnosis will be exactly defined with respect to the set of minimal global diagnoses. This definition

will be used in the next section, where an algorithm is designed that efficiently computes a unique set of minimal condensed diagnoses in each agent.

A condensed diagnosis should include the components and inputs that an agent $A_i$ use, which here are the common components $\mathcal{G}$, the private components for the specific agent $\mathcal{P}^{A_i}$, and the signals that not are outputs from the specific agent $\mathcal{S} \backslash \mathcal{OUT}^{A_i}$. Assume for example that $\{A, E, G\}$ is a global diagnosis, $A$ is a private component in agent $A_1$, $E$ is a private component in some other agent, and $G$ is a common component. A condensed diagnosis representing this global diagnosis in agent $A_1$ should include components $A$ and $G$, while component $E$ could be removed, resulting in the set $\{A, G\}$. However, to gain a correct cardinality of the condensed diagnosis, the information that one additional component did exist in the global diagnosis should be preserved. Here, the condensed diagnosis will therefore be represented by a tuple $\langle D, k \rangle$, where the set $D$ includes the used components and signals, and the scalar variable $k$ is the number of removed components. The variable $k$ is important since it preserves the cardinality of the global diagnoses. For example, the condensed diagnosis $\langle \{A, G\}, 1 \rangle$ represents the global diagnosis $\{A, E, G\}$.

Assume now that an agent has the condensed diagnoses $\langle \{A\}, 2 \rangle$ and $\langle \{B\}, 3 \rangle$, where both $A$ and $B$ are used by the agent. If a technician should repair the abnormally behaving agent, then it is interesting to know that component $A$ and $B$ should be checked since at least one of them are behaving abnormally and are affecting the agents behavior. It is not necessary to know exactly which other components in the other agents that are behaving abnormally. However, it is important to know that the condensed diagnosis including component $A$ has a lower cardinality, in this case $3 = |\{A\}| + 2$. While the condensed diagnosis including component $B$ has a cardinality of $4 = |\{B\}| + 3$. Based on the cardinality, the repair should start by checking component $A$ since the cardinality of the corresponding global diagnosis are lower than the global diagnosis including component $B$.

A condensed diagnosis is formally defined as follows.

DEFINITION 1 (Condensed diagnosis): *Let $\mathcal{D}$ be a set of minimal global diagnoses, where, for each diagnosis $\bar{D} \in \mathcal{D}$, $\bar{D} \subseteq \mathcal{C}$. The tuple $\langle D, k \rangle$, where $D \subseteq \mathcal{P}^{A_i} \cup \mathcal{G} \cup (\mathcal{S} \setminus \mathcal{OUT}^{A_i})$ and $k \in \mathbb{Z}$, is a condensed diagnosis in agent $A_i$ if and only if $\bar{D} \in \mathcal{D}$ exists such that*

$$
\begin{aligned}
&\text{a)} \quad |D| + k = |\bar{D}| \\
&\text{b)} \quad D \cap \mathcal{P} = \bar{D} \cap \mathcal{P}^{A_i} \\
&\text{c)} \quad D \cap \mathcal{G} = \bar{D} \cap \mathcal{G} \\
&\text{d)} \quad D \cap \mathcal{S} = \{s \in \mathcal{S} \setminus \mathcal{OUT}^{A_i} : \mathit{Dep}(s) \cap \bar{D} \neq \emptyset\}.
\end{aligned}
$$

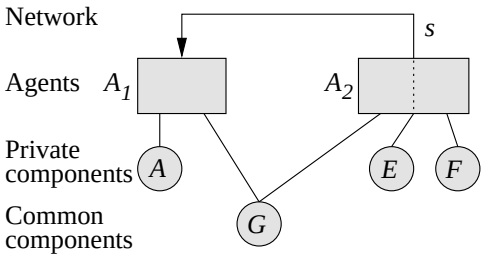



FIGURE 2: An example of a system consisting of two agents, four components A, E, F, and G, and one signal s. The agents use the components connected with lines.

Interpretation of the different requirements for a condensed diagnosis: a) means that the cardinality of D plus k should equal the cardinality of the global diagnosis; b) means that D should only include private components used by agent $A_i$; c) means that all common components should be included; d) means that signals, that might be abnormal due to the dependency on some abnormal component, should be included in D.

The following example will illustrate how a global diagnosis can be represented by a unique condensed diagnosis in two different agents.

EXAMPLE 2: Consider the system shown in Figure 2. A signal s exists whose dependency is $\mathtt{Dep}(s) = \{E\}$, represented by the dotted line. The sets of private components are $\mathcal{P}^{A_1} = \{A\}$ and $\mathcal{P}^{A_2} = \{E, F\}$. The set of common components is $\mathcal{G} = \{G\}$. The components connected by lines are used by the corresponding agent. Let $\{A, E, F, G\}$ be a minimal global diagnosis. A condensed diagnosis in agent $A_1$ is $\langle\{A, G, s\}, 1\rangle$, where component A is included since it is a private component in $A_1$, G since it is a common component, and s since it depends on the abnormal component E. Component F is represented by $k = 1$.

Agent $A_2$ is not interested in the same components as agent $A_1$, therefore, in this agent, the condensed diagnosis $\langle\{E, F, G\}, 1\rangle$ represents the global diagnosis. Here, component A has been removed and is represented by $k = 1$. ◇

In this simple example, the condensed diagnoses were not a very compact representation of the global diagnosis since a large part of the components is directly or indirectly supervised by both agents. However, for most distributed systems the number of the components that is supervised by several agents is not so high and the condensed diagnoses then become a compact representation of the global diagnoses. This is for example the case for the automotive application studied in the end of this paper.

Why should a condensed diagnosis include the signals instead of propagating these signals to the components on which they depends? The reason for this is that a signal might depend on many components that are not directly used by the agent that use the signal. This is for example the case for the automotive application, where the fluid amount signal depends on several components used only by the emission control system, such as the fluid injector and the fluid temperature sensor. Since these components are not directly used by the other agents, it is sufficient that the agent using a signal is aware of the abnormal behavior of the signal, and exactly which components that might have caused this abnormal behavior of the signal could be ignored.

According to the definition, the condensed diagnoses should include all signals $\mathcal{S}$ except those that are outputs from the current agent, $\mathcal{OUT}^{\mathcal{A}_i}$. An alternative here would be to limit this set to the smaller set of signals that are inputs to the current agent, $\mathcal{IN}^{\mathcal{A}_i}$. However, this would make the presentation of the algorithm in the next section more complex. Therefore, for clarity, a condensed diagnosis is defined with respect to the larger set of signals $\mathcal{S} \setminus \mathcal{OUT}^{\mathcal{A}_i}$.

Similar to minimal diagnoses, the minimal condensed diagnoses can be used to represent all condensed diagnoses.

DEFINITION 2 (Minimal condensed diagnosis): *A condensed diagnosis* $t = \langle D, k \rangle$ *is a* minimal condensed diagnosis *if and only if a condensed diagnosis* $t' = \langle D', k' \rangle$ *does not exist such that* $D' \subset D$ *and* $|D'| + k' < |D| + k$.

The set of minimal condensed diagnoses in one agent represents the set of minimal global diagnoses. In the following sections, an algorithm that computes all sets of minimal condensed diagnoses in each agent is designed and applied to an automotive application.

## 5 COMPUTING THE SETS OF MINIMAL CONDENSED DIAGNOSES

The set of minimal global diagnoses can be computed from all conflicts in all agents. However, the set of minimal global diagnoses can also be computed by merging the sets of local diagnoses in the different agents, see Section 4.1. This idea of computing the global diagnoses by merging the sets of local diagnoses, will be used in this section to construct an algorithm that merges the local diagnoses that include components or signals used by the other agents, such that the result is a set of minimal condensed diagnoses in each agent.

The outline of the algorithm is that each agent first computes its set of minimal local diagnoses, and then transmits, to all other agents, the minimal local diagnoses that include components used by other

agents. After this, each agent receives the sets of diagnoses transmitted from the other agents and then merges the received sets with its own set of minimal local diagnoses that results in the set of minimal condensed diagnoses.

Assume for example that agent $A_1$ has the minimal local diagnosis $\{A, G\}$ where $A$ is a private component and $G$ is a common component. Assume further that $A_2$ has the minimal local diagnosis $\{E, G\}$ where $E$ is a private component. To compute the minimal condensed diagnoses, the algorithm designed in this section transmits the tuple $\langle\{G\}, 1\rangle$ from agent $A_1$ to agent $A_2$. This tuple, which has the same structure as a condensed diagnosis, includes the part of the local diagnosis in agent $A_1$ that is used by the other agent. This transmitted tuple is merged with the minimal local diagnosis $\{E, G\}$ in $A_2$ that give the minimal condensed diagnosis $\langle\{E, G\}, 1\rangle$. As should be, this coincides with the minimal condensed diagnosis computed from the minimal global diagnosis $\{A, E, G\}$. Similarly, to compute the minimal condensed diagnosis in agent $A_1$, agent $A_2$ transmits the tuple $\langle\{G\}, 1\rangle$, which is merged with the minimal local diagnosis in $A_1$ resulting in the minimal condensed diagnosis $\langle\{A, G\}, 1\rangle$. Also the minimal condensed diagnosis in $A_1$ coincides with the minimal global diagnosis.

The transmitting is more fully described in Section 5.1, the receiving and merging are described in Section 5.2, and finally the main algorithm is described in Section 5.3. In the algorithms, $D$ is a diagnosis, $\Gamma \subseteq \mathcal{S}$, $\Omega \subseteq \mathcal{OUT}^{A_i}$, $P \subseteq \mathcal{P}$, and finally $G \subseteq \mathcal{G}$.

## 5.1 *Transmit the Interesting Local Diagnoses to All Agents*

Using the definition of condensed diagnosis, it can be seen that a minimal local diagnosis in one agent is of interest for the other agents if it includes components or signals used by some agent, or a component that some signal depends on. The local diagnoses including such components or signals should be transmitted to the other agents. However, before the diagnoses are transmitted, the private components can be removed since these are not used by any other agent. Consider for example an agent $A_1$ that has computed the local diagnosis $\{A, B, G\}$, where components $A$ and $B$ are private components in $A_1$, $G$ is a common component, and $s$ is a signal depending on $B$. Components $A$ and $B$ can be removed from the diagnosis when it is transmitted to the other agents since these are private, resulting in the set $\{G\}$. However, if a signal depends on any of the removed private components, then this signal is abnormal. Since some other agent might use this signal, it should be added to the set. In the example, this is the case for the signal $s$ that depends on the component $B$ included in the local diagnoses. Replacing the component with the depending signal results in the set $\{G, s\}$.

**Algorithm 1** – `Transmit`$(\mathbb{D}^{A_i})$. Transmit the interesting minimal local diagnoses.

**Input:** A set of minimal local diagnoses $\mathbb{D}^{A_i}$.
**Output:** Set of tuples $TX^{A_i}$ representing the interesting diagnoses in $\mathbb{D}^{A_i}$.

1: $\mathbb{D}^{TX} := \{D \in \mathbb{D}^{A_i} : D \cap (\mathcal{S} \cup \mathcal{G} \cup (\cup_{s \in \mathcal{S}} \mathtt{Dep}(s))) \neq \emptyset\}$
2: $TX^{A_i} := \emptyset$
3: **for each** $D = P \cup G \cup \Gamma \in \mathbb{D}^{TX}$ **do**
4: $\quad \Omega := \{s \in \mathcal{S} : \mathtt{Dep}(s) \cap P \neq \emptyset\}$
5: $\quad k := |P| - |\Omega|$
6: $\quad TX^{A_i} := TX^{A_i} \cup \{\langle G \cup \Gamma \cup \Omega, k\rangle\}$
7: **end for**
8: $TX^{A_i} := \mathtt{RemoveNonMinimal}(TX^{A_i})$
9: **if** $\mathbb{D}^{A_i} \setminus \mathbb{D}^{TX} \neq \emptyset$ **then**
10: $\quad k := \min_{D \in \mathbb{D}^{A_i} \setminus \mathbb{D}^{TX}} |D|$
11: $\quad TX^{A_i} := TX^{A_i} \cup \{\langle \emptyset, k\rangle\}$
12: **end if**

A problem when transmitting the set $\{G, s\}$ is that it is not known which cardinality that the resulting condensed diagnosis should have. The reason for this is that the set $\{G, s\}$ might represent different local diagnoses, such as $\{A, B, G\}$ or $\{A, C, D, G\}$. Therefore, to gain a correct cardinality for the resulting condensed diagnosis, the variable $k$, whose value is the number of removed private components minus the number of added signals, should be included when the set is transmitted to the other agents. If $D$ is the set to be transmitted, then the set plus the variable $k$ is give a tuple $\langle D, k\rangle$, which is similar to a condensed diagnosis. In the example, this results in the set $\langle\{G, s\}, 1\rangle$.

The minimal local diagnoses that are not transmitted to the other agents can be represented by a tuple including the empty diagnosis and the variable $k$ whose value is the minimal cardinality of any of the not transmitted minimal local diagnoses, i.e. a tuple $\langle\emptyset, k\rangle$. The agents receiving this tuple will then be aware of that one or more non-transmitted minimal local diagnoses with cardinality $k$ or higher exist. In the example, if also a minimal local diagnosis $\{A, C\}$ exists in agent $A_1$, then this local diagnosis would be represented by the tuple $\langle\emptyset, 2\rangle$.

Algorithm 1 performs the steps described above. Row 1 decides which minimal local diagnoses that include components or signals used by other agents, resulting in the set $\mathbb{D}^{TX}$. Row 2–7 construct a tuple for each local diagnosis $D = P \cup G \cup \Gamma$ in the set $\mathbb{D}^{TX}$, where the set $P$ is the private components, $G$ is the common components, and $\Gamma$ is the signals included in the local diagnosis. Each tuple in the set $TX^{A_i}$ includes a set consisting of $G$ and $\Gamma$, which are taken directly

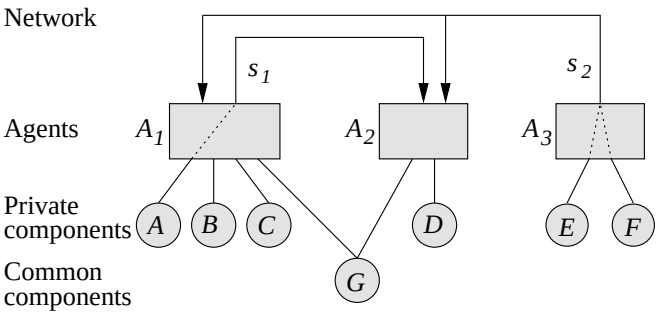



FIGURE 3: An example of a system consisting of three agents, seven components, and two signals. The components connected with lines are used by the corresponding agents.

from the local diagnosis, and the set $\Omega$ that consists of the signals that depend on any of the removed private components in $P$. Each tuple also includes the variable $k$, which equals the number of removed private components in $P$ minus the number of added signals in $\Omega$. In row 8, the function `RemoveNonMinimal`$(\cdot)$ removes all non-minimal tuples, where a minimal tuple is defined in the same way as a minimal condensed diagnosis, see Definition 2. Finally, Row 9–12 adds a tuple representing the not transmitted minimal local diagnoses.

EXAMPLE 3: Consider the system shown in Figure 3, where the sets of private components are $\mathcal{P}^{A_1} = \{A, B, C\}$, $\mathcal{P}^{A_2} = \{D\}$, and $\mathcal{P}^{A_3} = \{E, F\}$, while the set of common components is $\mathcal{G} = \{G\}$. Two signals exist with dependencies $\texttt{Dep}(s_1) = \{A\}$ and $\texttt{Dep}(s_2) = \{E, F\}$. Assume that the following set of minimal local diagnoses is computed in agent $A_1$

$$\mathbb{D}^{A_1} = \{\{B, s_2\}, \{A, B\}, \{G, B\}, \{C\}\}.$$

Using Algorithm 1, the set of local diagnoses that are of interest for the other agents is first computed,

$$\mathbb{D}^{TX} = \{\{B, s_2\}, \{A, B\}, \{G, B\}\}.$$

The private components are removed and the set of tuples to transmitted is thereby

$$TX^{A_1} = \{\langle\{s_2\}, 1\rangle, \langle\{s_1\}, 1\rangle, \langle\{G\}, 1\rangle, \langle\emptyset, 1\rangle\}$$

where the tuple $\langle\emptyset, 1\rangle$ represents the non-transmitted diagnosis $\{C\}$. The set $TX^{A_1}$ will represent the set of local diagnoses in $A_1$ when the agents $A_2$ and $A_3$ are computing their unique set of minimal condensed diagnoses. $\diamond$

**Algorithm 2** – `ReceiveCondense`$(\{TX^{A_1}, \ldots, TX^{A_n}\})$. Compute the minimal condensed diagnoses in agent $A_i$.

**Input:** For each agent $A_j$ except $A_i$, a received set $TX^{A_j}$ resulting from the evaluation of `Transmit`$(\cdot)$. The set of minimal local diagnoses $\mathbb{D}^{A_i}$.

**Output:** The set of minimal condensed diagnoses $\mathbb{D}_c^{A_i}$.

1: **for each** $A_j$ except $A_i$ **do**
2: $\quad RX^{A_j} := \emptyset$
3: $\quad$ **for each** $\langle G \cup \Gamma \cup \Omega, k\rangle \in TX^{A_j}$ **do**
4: $\quad\quad \bar{\Omega} := \Gamma \cap \mathcal{OUT}^{A_i}$ and $\bar{\Gamma} := \Gamma \setminus \bar{\Omega}$
5: $\quad\quad$ **for each** $\bar{P} \in$ `CrossProduct`$(\cup_{s\in\bar{\Omega}}\{$`Dep`$(s)\})$ **do**
6: $\quad\quad\quad RX^{A_j} := RX^{A_j} \cup \{\langle \bar{P} \cup G \cup \bar{\Gamma} \cup \bar{\Omega}, k\rangle\}$
7: $\quad\quad$ **end for**
8: $\quad$ **end for**
9: **end for**
10: $RX^{A_i} := \{\langle D, \emptyset\rangle : D \in \mathbb{D}^{A_i}\}$
11: $\mathbb{D}_c^{A_i} :=$ `CrossProductTuple`$(\{RX^{A_1}, \ldots, RX^{A_n}\})$
12: $\mathbb{D}_c^{A_i} :=$ `RemoveNonMinimal`$(\mathbb{D}_c^{A_i})$

## 5.2 *Receive and Merge the Transmitted sets of Tuples*

The second step when calculating the sets of minimal condensed diagnoses is for each agent to receive the sets of tuples transmitted from the other agents, transform them into an appropriate form, and then compute the minimal condensed diagnoses. If a received tuple includes a signal $s$ that is an output from the receiving agent then the receiver is interesting in knowing which private components that could have caused the abnormal behavior of the signal. The signal is therefore replaced with the components that it depends on. If for example the tuple $\langle\{s\}, 0\rangle$ is received and $s$ is an output from the receiving agent with dependency `Dep`$(s) = \{E, F\}$, then this tuple is replaced with the tuples $\langle\{E\}, 0\rangle$ and $\langle\{F\}, 0\rangle$, since either is component $E$ abnormal or component $F$. After all such signals have been replaced by their dependencies, the minimal local diagnoses in the receiving agent and the sets of received tuples are merged and this results in the set of minimal condensed diagnoses.

Algorithm 2 performs the steps described above. Row 1–9 transform each set of transmitted tuples, such as $TX^{A_j}$ transmitted by agent $A_j$, into sets of received tuples, such as $RX^{A_j}$. For each tuple $\langle G \cup \Gamma \cup \Omega, k\rangle$ in the set $TX^{A_j}$, row 4 computes a set $\bar{\Omega}$ that consists of the signals in $\Gamma$ that are outputs from the receiving agent $A_i$, and the set $\bar{\Gamma}$ that are the signals in $\Gamma$ except those in $\bar{\Omega}$. The signals in $\bar{\Omega}$ are then replaced by a set consisting of one component from the dependency

of each signal, resulting in a set of components $\bar{P}$. The received tuple $\langle \bar{P} \cup G \cup \bar{\Gamma} \cup \Omega, k \rangle$ is thereafter constructed and stored in the set of tuples $RX^{A_j}$. The computation of each set $\bar{P}$ is done in row 5 using the function $\texttt{CrossProduct}(M)$ that computes the cross product for the collection of sets $M$. If for example a signal $s_1$ has a dependency $\texttt{Dep}(s_1) = \{A, B\}$ and another signal $s_2$ has a dependency $\texttt{Dep}(s_2) = \{C\}$, then the set of signals $\bar{\Omega} = \{s_1, s_2\}$ is replaced with a set in the set

$$\texttt{CrossProduct}(\{\texttt{Dep}(s_1), \texttt{Dep}(s_2)\}) = \{\{A, C\}, \{B, C\}\}.$$

Notice that one received tuple in the set $TX^{A_j}$ could result in several received tuples in the set $RX^{A_j}$. This is for example the case for the tuple $\langle \{s_1, s_2\}, k \rangle$, where the signals have the dependencies stated above. This tuple would give the two tuples $\langle \{A, C\}, k \rangle$ and $\langle \{B, C\}, k \rangle$ due to the dependencies of the signals.

Row 10 transforms the minimal local diagnoses in agent $A_i$ to the same format as the received tuples. In row 11–12 in the algorithm, the sets $RX^{A_1}, \ldots, RX^{A_n}$ are merged and the non-minimal condensed diagnoses are removed. The function $\texttt{CrossProductTuple}(M)$ for a set $M$ of sets of tuples performs a normal cross product with respect to the sets included in the tuples and a summation of the variables $k$. For example,

$$\texttt{CrossProductTuple}\big(\{\{\langle D_1, k_1 \rangle\}, \{\langle D_{21}, k_{21} \rangle, \langle D_{22}, k_{22} \rangle\}\}\big) = \\ \big\{\langle D_1 \cup D_{21}, k_1 + k_{21} \rangle, \langle D_1 \cup D_{22}, k_1 + k_{22} \rangle\big\}.$$

The function $\texttt{RemoveNonMinimal}(\cdot)$ removes all non-minimal condensed diagnoses.

In an efficient implementation, row 11–12 should probably be replaced with an operation similar to a minimal hitting set operation. Such an operation would directly compute the set of minimal condensed diagnoses, instead of first computing the often larger set of condensed diagnoses that results from the cross-product, and then remove the non-minimal condensed diagnoses.

## 5.3 *Main Algorithm for the Computation of the Sets of Minimal Condensed Diagnoses*

Based on the algorithms designed above, Algorithm 3 can be used to compute the set of minimal condensed diagnoses in each agent. For each agent, the algorithm computes the set of minimal local diagnoses $\mathbb{D}^{A_i}$ in row 1 using a minimal hitting set algorithm, see Section 3. The algorithm then evaluates row 2–5 that includes calls to Algorithm 1, where each call has a set of minimal local diagnoses as input and give

**Algorithm 3** – `Main`. Computation of the set of minimal condensed diagnoses in each agent.

**Input:** The set of conflicts $\Pi^{A_i}$ in each agent $A_i$.
**Output:** The set of minimal condensed diagnoses $\mathbb{D}_c^{A_i}$ in each agent $A_i$.

1: **for each** agent $A_i$ **do** $\mathbb{D}^{A_i} :=$ `MinimalHittingSets`$(\Pi^{A_i})$ in $A_i$
2: **for each** Agent $A_i$ **do**
3: compute $TX^{A_i} :=$ `Transmit`$(\mathbb{D}^{A_i})$ in agent $A_i$
4: broadcast $TX^{A_i}$ on the network
5: **end for**
6: **for each** Agent $A_i$ **do**
7: receive $TX^{A_j}$ in $A_i$ from all other agents $A_j$ except $A_i$
8: compute $\mathbb{D}_c^{A_i} :=$ `ReceiveCondense`$(\{TX^{A_1}, \ldots, TX^{A_n}\} \backslash TX^{A_i})$ in $A_i$
9: **end for**

a set of transmitted diagnoses $TX^{A_i}$ as output. Finally are row 6–9 evaluated that includes calls to Algorithm 2, where each call has the set of transmitted sets $TX^{A_1}$, ..., $TX^{A_n}$ except $TX^{A_i}$ as input. After that Algorithm 3 has been evaluated, each agent has a unique set of minimal condensed diagnoses $\mathbb{D}_c^{A_i}$.

EXAMPLE 4: This example uses the same system as is used in Example 3. Algorithm 3 first computes the minimal local diagnoses in each agent and then calls Algorithm 1 that, for each agent, computes the set of tuples that should be transmitted to the other agents. Assume that agent $A_1$ has received the following sets from agent $A_2$ and $A_3$,

$$TX^{A_2} = \{\langle\{s_1\}, 0\rangle, \langle\emptyset, 1\rangle\} \qquad TX^{A_3} = \{\langle\{s_2\}, 0\rangle\}.$$

Algorithm 3 calls Algorithm 2 in agent $A_1$, where the sets of received tuples are transformed into the sets

$$RX^{A_2} = \{\langle\{A\}, 0\rangle, \langle\emptyset, 1\rangle\} \qquad RX^{A_3} = \{\langle\{s_2\}, 0\rangle\}.$$

Assume again that $A_1$ has the following set of minimal local diagnoses

$$\mathbb{D}^{A_1} = \{\{B, s_2\}, \{A, B\}, \{G, B\}, \{C\}\}.$$

The set of minimal local diagnoses is transformed by Algorithm 2 to

$$RX^{A_1} = \{\langle\{B, s_2\}, 0\rangle, \langle\{A, B\}, 0\rangle, \langle\{G, B\}, 0\rangle, \langle\{C\}, 0\rangle\}.$$

The sets $RX^{A_1}$, $RX^{A_2}$, and $RX^{A_3}$ are then merged, resulting in the set

$$\mathbb{D}_c^{A_1} = \{\langle\{A, B, s_2\}, 0\rangle, \langle\{s_2, B\}, 1\rangle, \langle\{s_2, C\}, 1\rangle\}, \langle\{A, s_2, C\}, 0\rangle\}$$

that is the set of minimal condensed diagnoses in $A_1$. ◇

What is the interpretation of the set of minimal condensed diagnoses in Example 4? The first condensed diagnosis for example states that the local components A and B, and the signal $s_2$ could explain the abnormal behavior of the system. It is not exactly known which component that has caused the abnormal behavior of the signal, and agent $A_1$ is not interested in knowing this. It is sufficient to know that the signal $s_2$ that $A_1$ use might be abnormal. The second diagnosis states that the abnormal behavior of the component B, the signal $s_2$, and some other component used in some other agent could also explain the abnormal behavior of the system. Both of these diagnoses have the same cardinality, it is therefore not possible to conclude that a repair technician should check one of them before the other. If the variable k had not been included in the condensed diagnoses, then the cardinality of the second diagnosis would be wrongly be two, which is one lower than the first diagnosis.

## 5.4 *Minimal Cardinality Condensed Diagnoses*

In some cases, the search for diagnoses is focused onto some smaller set of diagnoses, such as the set of minimal cardinality diagnoses. The cardinality of a diagnosis is the number of components included in the diagnosis. The reason why the diagnoses with minimal cardinality are of interest is that they point at the more likely diagnoses, considering the number of components included in the diagnoses. If for example the two diagnoses $\{B\}$ and $\{A, C\}$ have been computed, and the a-priori probability that a component is abnormal is small and approximately of the same magnitude for all components, then it is more probable that component B is abnormal than that component A and C are both abnormal. The most probable diagnosis $\{B\}$ is in this case also the minimal cardinality diagnosis.

Considering condensed diagnoses, then the set of minimal cardinality condensed diagnoses can directly be computed from the set of minimal condensed diagnoses. However, this approach is not efficient since it often requires the computation of many condensed diagnoses that are not minimal cardinality condensed diagnoses. Therefore, algorithms designed above will in this section be extended such that they directly compute the set of minimal cardinality condensed diagnoses in each agent.

To extend the algorithms such that the minimal cardinality condensed diagnoses are computed, insert a new row

12: $\mathbb{D}_c^{A_i} := \{\langle D, k\rangle \in \mathbb{D}_c^{A_i} : |D| + k = \min_{\langle \bar{D}, \bar{k}\rangle \in \mathbb{D}_c^{A_i}} |\bar{D}| + \bar{k}\}$

before row 12 in Algorithm 2. The new row removes the condensed diagnoses that are not minimal cardinality. Further, let Algorithm 3 include an input $\mathbb{A}$ that is a set of agents, and change row 6 to

6: **for each** agent $A_i \in \mathbb{A}$ **do**

such that row 6–9 are only evaluated for the agents in the set $\mathbb{A}$. Finally, remove row 1 in Algorithm 3.

After the changes described above have been implemented, Algorithm 4 can be used to compute the set of minimal cardinality condensed diagnoses in each agent. The algorithm uses the function `Main`, i.e. Algorithm 3. Algorithm 4 first calculates the set of minimal local diagnoses, stored as $X^{\mathcal{A}_i}$, in each agent. For each agent, the set of diagnoses $\mathbb{D}^{\mathcal{A}_i}$, used by `Main`, is defined such that at least one minimal local diagnosis with minimal cardinality will be transmitted from each agent when `Main` is called. After the call to `Main`$(\{A_1\})$, agent $A_1$ has a set of condensed diagnoses. It is possibly to change agent $A_1$ to any other agent, for example an agent with free processing power.

Using the condensed diagnoses in agent $A_1$, an upper limit $m$ on the cardinality of the condensed diagnoses with minimal cardinality can be computed. Using this limit, the set $\mathbb{D}^{\mathcal{A}_i}$ is redefined such that only the minimal local diagnoses with cardinality lower than this limit is included in the set. The local diagnoses with cardinality higher than $m$ can not be part of a minimal cardinality condensed diagnosis and can therefore be ignored. With this new set $\mathbb{D}^{\mathcal{A}_i}$, all condensed diagnoses with minimal cardinality will be calculated when `Main`$(\mathcal{A})$ is called with the set of all agents as input.

Even though Algorithm 4 is written as two separated parts, row 1–3 and 4–6, the result of the first part should in an efficient implementation probably be used when evaluating part two. The parts that can be reused are primely the sets of tuples that have been transmitted when `Main` is called in row 3.

---

**Algorithm 4** – `MainMinimalCardinality` Computation of the sets of minimal cardinality condensed diagnoses.

---

**Input:** The set of conflicts $\Pi^{\mathcal{A}_i}$ in each agent $A_i$.

**Output:** A set of minimal cardinality condensed diagnoses $\mathbb{D}^{\mathcal{A}_i}_{s,mc}$ in each agent $A_i$.

1: **for each** agent $A_i$ **do** $X^{\mathcal{A}_i} :=$ `MinimalHittingSets`$(\Pi^{\mathcal{A}_i})$ in $A_i$
2: **for each** agent $A_i$ **do** $\mathbb{D}^{\mathcal{A}_i} = \{D \in X^{\mathcal{A}_i} : |D| = \min_{\bar{D} \in X^{\mathcal{A}_i}} |\bar{D}|\}$ in $A_i$
3: `Main`$(\{A_1\})$ [*Agent* $A_1$ *has a set of condensed diagnoses,* $\mathbb{D}^{\mathcal{A}_1}$]
4: $m = \min_{\langle D,k\rangle \in \mathbb{D}^{\mathcal{A}_1}_c} |D| + k$ , computed in $A_1$
5: **for each** agent $A_i$ **do** $\mathbb{D}^{\mathcal{A}_i} = \{D \in X^{\mathcal{A}_i} : |D| = m\}$ in $A_i$
6: **for each** agent $A_i$ **do** $\mathbb{D}^{\mathcal{A}_i}_{s,mc} :=$ `Main`$(\mathcal{A})$

---

# 6 AUTOMOTIVE VEHICLE APPLICATION

The design of Algorithm 3 is motivated by efficiency compared to computing the complete set of minimal global diagnoses in a central agent, and in the reduction of the number and sizes of the minimal condensed diagnoses compared to the minimal global diagnoses. To verify these motivations, the algorithm is in this section applied to an automotive vehicle and compared to a centralized implementation that computes the set of minimal global diagnoses. Algorithm 4, which computes the sets of minimal cardinality condensed diagnoses, is also applied to the automotive vehicle, and the results are given later in Section 7.

## *6.1 Test Cases Used in the Application*

Algorithm 3 is here applied to a part of the diagnostic system used in the Scania heavy duty vehicle described in Section 2. The part that is used includes the engine management system (EMS), the selective catalytic reduction system (SCR), and the part of the coordinator system (COO) affecting the EMS and the SCR. The EMS agent $A_{EMS}$ controls and supervises the engine, the SCR agent $A_{SCR}$ controls and supervises the system used to reduce nitrous oxides, while the COO agent $A_{COO}$ makes a temperature sensor value available on the network for use by the other agents. There are 85 components, with components numbered 1 to 4 being common, components 5 to 50 being private in $A_{EMS}$, components 51 to 84 being private in $A_{SCR}$, and component 85 being private in $A_{COO}$. There are three signals of which two are outputs from $A_{EMS}$ and one is an output from the $A_{COO}$. Figure 4 shows a schematic overview of the system, using the same notation as used in the previous examples. Both the $A_{EMS}$ and the $A_{SCR}$ use and supervise the signals from $A_{COO}$, while $A_{SCR}$ uses and supervises the two signals from the $A_{EMS}$.

The diagnostic systems in the $A_{EMS}$ and the $A_{SCR}$ consist of 57 and 55 diagnostic tests respectively. The isolation structures can be seen in Figure 5 and 6 where a cross $\times$ in row $i$ and column $j$ means that the diagnostic test $i$ supervises component $j$ for abnormal behavior. The private component in $A_{COO}$ is not directly supervised, however, the signal $s_1$ from $A_{COO}$ depends on its private component and is therefore indirectly supervised by the other agents. Finally, the signals $s_2$ and $s_3$ from $A_{EMS}$ depend on the private component 5 and 6 respectively.

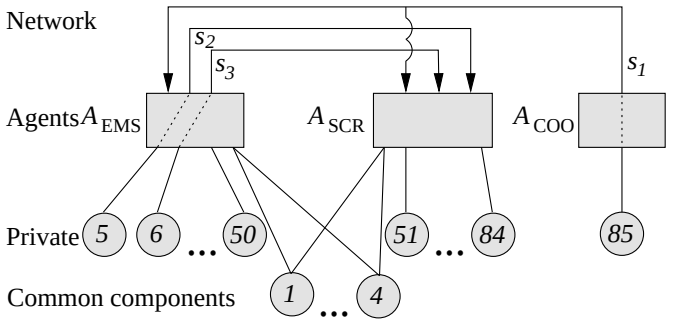



FIGURE 4: The distributed system used in the application.

## 6.2 *Computing the Minimal Global Diagnoses*

Algorithm 3 is compared to a centralized algorithm based on the method for computing minimal diagnoses from minimal conflicts given in (Kleer and Williams, 1987). In the centralized algorithm, all conflicts from all agents are transmitted to $A_{COO}$ where the set of minimal global diagnoses is computed. The main idea in the method given in (Kleer and Williams, 1987), is that the set of diagnoses is initialized to the empty diagnosis that then is updated such that it is consistent with every conflict. This is performed by checking if a diagnosis has an empty intersection with a conflict, and if this is not the case, then the diagnosis is extended to several new diagnoses where each new diagnosis includes the old diagnosis and one component from the conflict. The non-minimal extended diagnoses are then removed. The diagnoses are extended until all conflicts have been considered and the result is the set of minimal diagnoses.

## 6.3 *Operations and Transmissions*

To compare the designed algorithm and the centralized algorithm, the numbers of non-trivial transmissions and operations needed to compute the diagnoses are counted.

For each diagnosis or conflict that is transmitted on the network, the number of transmissions is assumed to be equal to the cardinality of the diagnosis or conflict. The operations that are considered non-trivial are described below.

For the centralized algorithm, and the computation of the minimal local diagnoses in Algorithm 3, one operation is needed to check if each diagnosis has an empty intersection with a conflict, and one operation is needed to extend each diagnosis. For Algorithm 1, one operation is needed to check if a local diagnosis should be transmit-

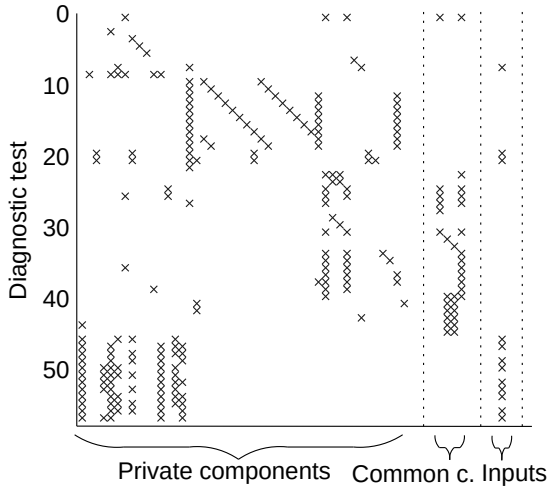



FIGURE 5: The isolation structure for the diagnostic system in the $A_{EMS}$.

ted, $|\mathcal{OUT}^{A_i}|$ operations to transform it to a tuple, one operation to check if one transmitted tuple is minimal considering one other transmitted tuple, and $|\mathbb{D}^{A_i} \setminus \mathbb{D}^{TX}|$ operations to get the tuple representing the non-transmitted local diagnoses. In Algorithm 2, two operations are needed in row 4. Row 5 and 11 requires one operation for each constructed set. Row 10 requires one operation for each local diagnosis. Finally, one operation is needed for each condensed diagnosis that is checked for non-minimality considering one other condensed diagnosis.

## 6.4 *Evaluation For One Test Case*

To evaluate the algorithms, components numbered 6 in $A_{EMS}$ and 75 in $A_{SCR}$ are affected such that they are behaving abnormally. It is here assumed that every diagnostic test that supervises an abnormal component has responded. With the given abnormal components, one test has responded in $A_{EMS}$ and two tests have responded in $A_{SCR}$.

Algorithm 3 is now used to compute the set of minimal condensed diagnoses in each agent. In $A_{EMS}$, the set of minimal condensed diagnoses is

$$\{\langle\{6\}, 1\rangle\},$$

in $A_{SCR}$ the set of minimal condensed diagnoses is

$$\{\langle\{s_3, 71\}, 0\rangle, \langle\{s_3, 73\}, 0\rangle, \langle\{s_3, 74\}, 0\rangle, \langle\{s_3, 75\}, 0\rangle, \langle\{s_3, 76\}, 0\rangle\},$$

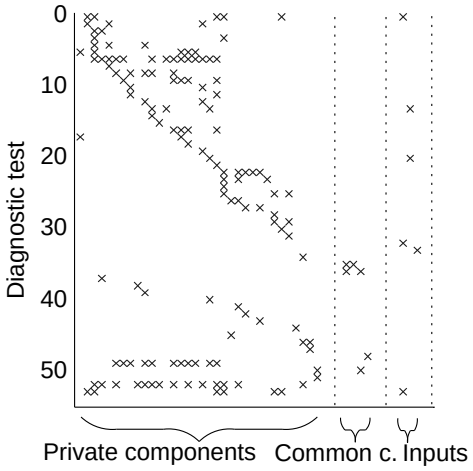



FIGURE 6: The isolation structure for the diagnostic system in the $A_{SCR}$.

and in $A_{COO}$ the set of minimal condensed diagnoses is

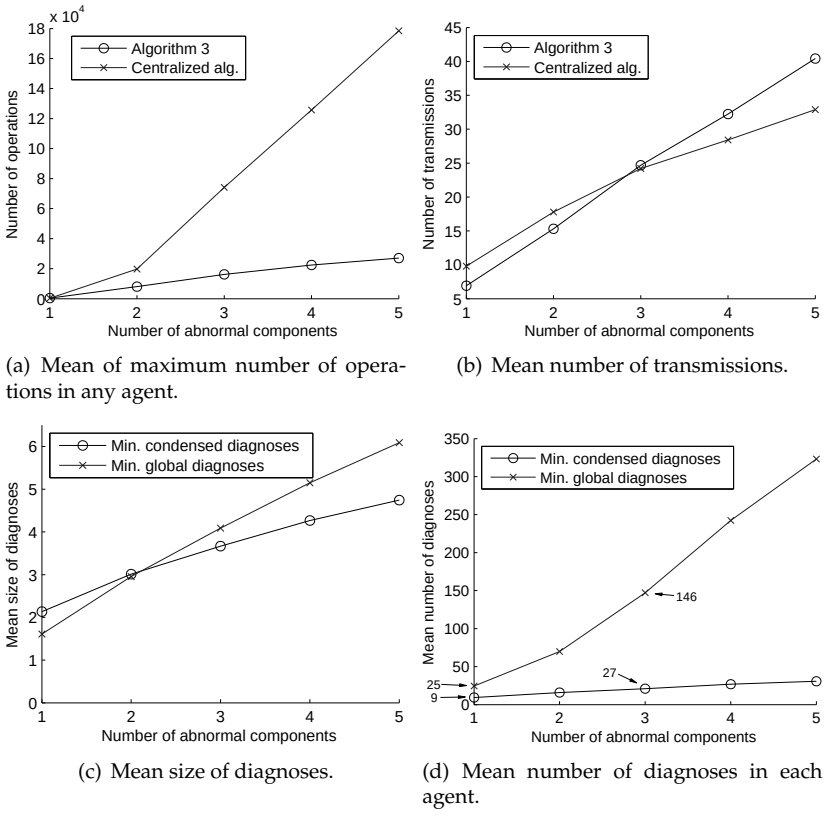
$$\{\langle\{s_3\}, 1\rangle\}.$$

It can be seen that only private components are included in the minimal condensed diagnoses, and that the signal $s_3$ that depends on the abnormal component numbered 6 is included in the $A_{SCR}$ and the $A_{COO}$. Using the centralized algorithm, the set of minimal global diagnoses has been computed resulting in the set

$$\{\{6, 71\}, \{6, 73\}, \{6, 74\}, \{6, 75\}, \{6, 76\}\}.$$

Considering the signals dependencies, it can be seen that the minimal global diagnoses coincides with the result from Algorithm 3 given above.

The gain in the reduction of the number of diagnoses when using Algorithm 3 are seen in $A_{EMS}$ and in $A_{COO}$ where there are only two minimal condensed diagnoses, while there are five minimal global diagnoses. This is no major reduction, however, as will be seen later in this section, the reduction in number of diagnoses is in other cases much larger. If for example the $A_{EMS}$ should perform fault tolerant control based on the diagnostic tests results, it is an advantage to use the condensed diagnosis since the condensed diagnosis $\langle\{6\}, 1\rangle$ includes all information in the global diagnoses that are of interest for this agent. Since only component numbered 6 is used by the $A_{EMS}$, it is not necessary to know that the components 71, 73, 74, 75, and 76 are also included in the set of minimal global diagnoses.




(a) Mean of maximum number of operations in any agent.

(b) Mean number of transmissions.

(c) Mean size of diagnoses.

(d) Mean number of diagnoses in each agent.

FIGURE 7: The figures show the mean number of operations, transmissions, diagnoses in each agent, and mean size of the diagnoses.

Further, the maximum number of needed operations in any agent is 144 for the centralized algorithm but only 65 for Algorithm 3, a reduction of 55 %. The number of needed transmissions is 8 for Algorithm 3 but 12 for the centralized algorithm.

## 6.5 *Evaluation for a Suite of Tests*

Above, the Algorithm 3 and the centralized algorithm were compared for one specific test case. To further compare the algorithms, a test suite is created by randomly make some components in the automotive system behave abnormally. The mean of the maximum number of needed operations in any agent can be seen in Figure 7(a). For one and two abnormal components, all combinations of components are used, while for the other sizes, five thousand different abnormal com-

ponents for each size of the set of abnormal components are used.

Figure 7(a) shows that for the test cases with more than one abnormal component, the mean of the maximum number of needed operations with Algorithm 3 is lower than the centralized algorithm, and it can also be seen that the growth rate is significantly lower. With for example two abnormal components, there is a reduction of the processor load with 50 % increasing to 85 % reduction for five abnormal components. Automotive applications can typically consist of 2000 or more components and if only a small part of these fail, two or more abnormal components could exist in the system at the same time. Further, a common case is that components are not immediately repaired, even though a fault is detected and isolated. A consequence of this is that the number of abnormal components increases over time.

The numbers of needed transmissions for the two algorithms are shown in Figure 7(b), where it can be seen that there is no significant difference between the algorithms.

One motivation for the introduction of the condensed diagnoses is to remove the non-used components from the global diagnoses. Two values that therefore are of interest are the mean size and mean number of diagnoses. If the minimal condensed diagnoses truly are a condensed representation of the minimal global diagnoses, then both these values should be reduced for the minimal condensed diagnoses. Notice that these numbers are a property of the minimal condensed diagnoses and the minimal global diagnoses, and are therefore independent of the algorithms that are used to compute the diagnoses.

Figure 7(c) shows the mean size of a diagnosis, where the size of a condensed diagnosis $\langle D, k \rangle$ is assumed to be $|D| + 1$, which should not be confused with its cardinality that is $|D| + k$. It can be seen in the figure, that the mean size of the diagnoses is reduced for the minimal condensed diagnoses when there are three or more abnormal components in the system. Further, the mean size of the global diagnoses has a higher growth rate than the condensed diagnoses.

In Figure 7(d), the mean number of minimal diagnoses per agent is shown. It can be seen that the number of minimal condensed diagnoses is significantly reduced compared to the number of minimal global diagnoses. It can for example be seen that there is a reduction of abut 70 % when two abnormal components exist in the system and increasing to over 90 % for five abnormal components. When there is one abnormal component there are, for this application, in mean 25 minimal global diagnoses but only 9 minimal condensed diagnoses. One could argue that since the system includes three agents that in mean have 9 diagnoses, the technician has to consider 27 diagnoses if all agents should be repaired. However, as can be seen in the figure, this argument only holds for one or two abnormal components.

For three abnormal components, there are a total of about 27×3=81 minimal condensed diagnoses, while there are 146 minimal global diagnoses. Considering repair, each of the diagnoses could direct the repair technician to the abnormally behaving components, and the repair task becomes more difficult when the number of diagnoses increases. The reduction in number of diagnoses therefore motivates the use of the minimal condensed diagnoses.

The memory usage needed to store the set of diagnoses is approximately the mean number of diagnoses times the mean size of the diagnoses. Primarily due to the reduction in number of diagnoses, the memory usage to store the sets of minimal condensed diagnoses is significantly reduced compared to the memory usage when the set of minimal global diagnoses is stored. For example is the memory usage reduced with 70 % for the automotive application when two components are behaving abnormally.

### *6.6 Conclusions from the Automotive Application*

The important conclusions from this application are that the designed algorithm is more efficient and requires a reduced number of transmissions compared to the computation of the set of minimal global diagnoses using a centralized algorithm. Further, both the mean size and the mean number of minimal condensed diagnoses are reduced compared to the minimal global diagnoses.

## 7 AUTOMOTIVE APPLICATION OF MINIMAL CARDINALITY CONDENSED DIAGNOSES

In some cases, the search for diagnoses is focused onto the set of minimal cardinality diagnoses, see Section 1, and in Section 5.4 an algorithm was designed that computes the sets of minimal cardinality condensed diagnoses in each agent. In this section, the algorithm is applied to the automotive application used in the previous section.

### *7.1 Implementation of the Minimal Cardinality Algorithm*

Here, the algorithm for the computation of the sets of minimal cardinality condensed diagnoses is implemented exactly as stated in Algorithm 4. This means that all results from the first part, i.e. row 2–3 in the algorithm, are discarded before evaluating the second part, i.e. row 4–6. The inefficiency caused by this give at most a doubling of the number of needed operations and transmissions. An implementation

that reuses the information would use similar ideas to those designed in Paper I, see Algorithm 2 in Section 3.5.

### 7.2 *Computing the Minimal Cardinality Global diagnoses*

Algorithm 4 is compared to a centralized algorithm similar to that used in Section 6.2, but which instead computes the set of minimal cardinality global diagnoses. Since only the set of minimal cardinality global diagnoses is wanted, the centralized algorithm is modified such that, after each extension of the diagnoses with respect to some conflict, the diagnoses with cardinality higher than the cardinality of the minimal cardinality global diagnoses are removed.

### 7.3 *Evaluation for a Test Suite*

The algorithms is compared in the same way as is done in Section 6 and the results are shown in Figure 8. When Algorithm 3 is compared to the centralized algorithm, it is found that the number of needed operations is lower for Algorithm 3 when two abnormal components or more exist in the system. Here, Algorithm 4 is in mean more efficient than the centralized algorithm when there are four abnormal components or more, and the gain increases when the number of abnormal components increases. For four components the gain is only 18 %, but the gain increases fast and reaches 70 % for five abnormal components. Further, the growth rate can be seen to be exponential for the centralized algorithm while only linear for Algorithm 4. Therefore, the gain in using Algorithm 4 increases exponentially with the number of abnormal components in the automotive application when four or more components are abnormal.

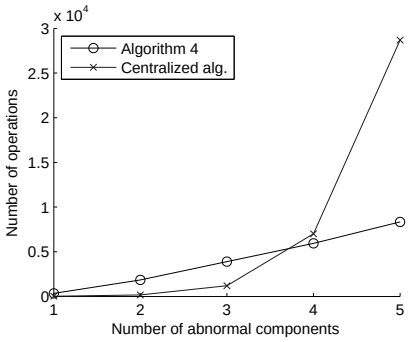
The number of needed transmissions can be seen in Figure 8(b). The number of operations increases somewhat faster for Algorithm 4 than for the centralized algorithm.

Figure 8(c) and 8(d) show the mean size and the mean number of the minimal cardinality condensed and global diagnoses. The results are similar to that in Section 6. There is for example a mean reduction of 35 % when one component is abnormal in the automotive application, increasing to a reduction of 73 % for five abnormal components.

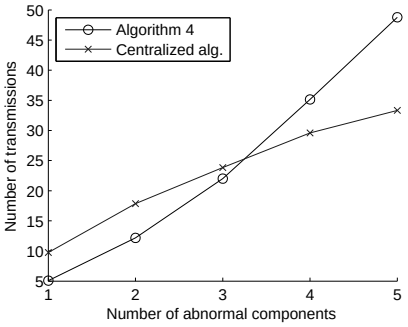
Especially, the sets of minimal cardinality condensed diagnoses are a compact representation of the minimal cardinality global diagnoses for the same reasons as stated in Section 6.

### 7.4 *Conclusions from the Automotive Application*

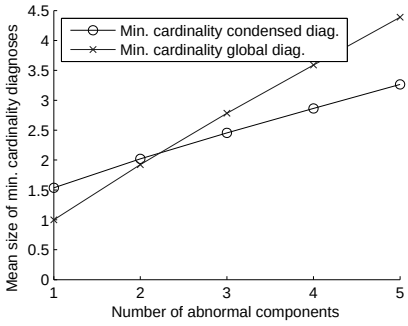
In this application, the gain in processor load when using Algorithm 4 grows exponentially for system with four our more abnormal compo-



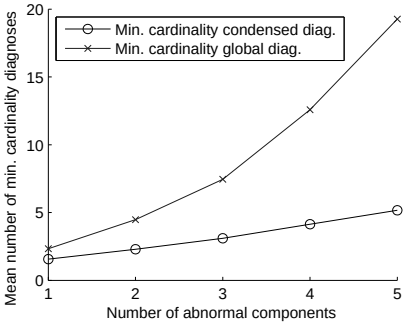

(a) Mean of maximum number of operations in any agent.




(b) Mean number of transmissions.




(c) Mean size of diagnoses.




(d) Mean number of diagnoses per agent.

FIGURE 8: The mean number of operations, transmissions, minimal diagnoses in each agent, and mean size of the diagnoses are shown.

nents. Independent of the algorithms, both the number and the size of the minimal cardinality condensed diagnoses are lower compared to the minimal cardinality global diagnoses.

Compared to the minimal condensed diagnoses and Algorithm 3 in Section 6, it can be seen that the mean number of minimal cardinality condensed diagnoses are lower than the number of minimal condensed diagnoses. This can for example be seen in Figure 7(d), where the number of minimal condensed diagnoses reaches about 25, and in Figure 8(d), where the number only reaches 5 minimal cardinality condensed diagnoses. Further, both the number of needed operations, transmissions, and the size of the minimal cardinality condensed diagnoses are reduced compared to the minimal condensed diagnoses.

# 8 EVALUATION FOR DETERMINISTIC SYSTEMS

In Paper I Section 4, each test case is exactly defined by a set of parameters that describes the characteristics of the diagnostic system. Since each test case is exactly defined by a set of parameters, it is possible to evaluate how the complexity of the algorithm in Paper I scales with the value of the parameters. By using these test cases also for Algorithm 3, the objective is to evaluate how the complexity, and the number and size of the minimal condensed diagnoses scales when the different parameters are changed.

To recapitulate, each test case consists of $n$ agents with $n_t$ diagnostic tests that have responded in each agent, thereby creating $n_t$ conflicts in each agent. Each test monitors $n_{c/t}$ components with an overlap of $n_{overlap}$ components between the tests. The agents are partitioned into sets of $n_{a/m}$ agents such that there is a connection of $n_{con}$ components between two sequential agents within each set and none between the sets. By these parameters, each test case is exactly defined. In the studied automotive vehicle, $n_{con}$ is commonly low compared to the number of supervised components, while $n_{overlap}$ might vary from 0 to $n_{c/t} - 1$.

## 8.1 *The Evaluated Algorithms*

The designed algorithms will be compared to a centralized algorithm similar to that in Paper I, Section 4. This centralized algorithm used here differs in that it here computes the complete set of minimal global diagnoses, in contrast to the set of minimal cardinality global diagnoses that is computed in Paper I. To reduce the complexity when using the designed algorithms and the centralized algorithm, both algorithms will partition the set of agents into modules. For the computation of the modules, the algorithm in Paper I Section 3.3 is used.

## 8.2 *Evaluation for a Test Suite*

Consider the test suite where $n_t = 3$, $n_{c/t} = 3$, $n_{overlap} = 1$, $n_{a/m} = 4$, $n_{con} = 1$, and $n$ is varied between one and twelve. The set of minimal condensed diagnoses and the set of minimal global diagnoses have been calculated using the designed algorithms and the centralized algorithm respectively, and the number of needed operations and transmissions have been counted. The results are shown in Figure 9.

Figure 9(a) and 9(b) show, in linear and logarithmic scale, the maximum number of operations in one agent and, for the designed algorithms, the sum of operations over all agents. Due to the number of agents per module, $n_{a/m} = 4$, the set of agents is partitioned into modules such that agent one to four is included in the first module,

five to eight in the second, etc. It can be seen that the maximum load increases exponentially when the size of the systems are increased from one to four agents since the first module includes up to $n_{a/m} = 4$ agents. The maximum number of needed operations equals that of the maximum number of operations in one of the modules, and thereby reaches a constant non-increasing value when there is is four or more agents in the system. The maximum load for the centralized algorithm increases linearly with the number of modules. The reduction in maximum processor load is for example reduced with 80 % for systems with two agents, increasing fast to 99.5 % reduction for the system with four agents.

For the test suite, the number of transmissions increases linearly with the number of modules for both algorithms , see Figure 9(c).

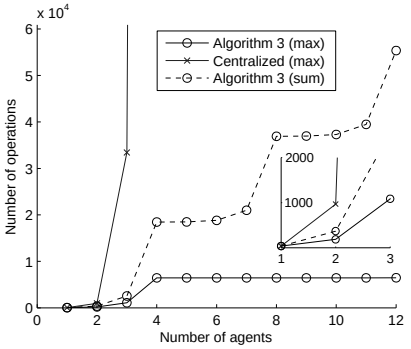
In Figure 9(d), the mean size of each diagnosis can be seen. The mean size increases for the first module but are thereafter almost constant. The reason for the constant value is that the diagnoses in one module are independent of the diagnoses in the other modules. The mean value of the size of the diagnoses are constant within each complete module, and the mean value over all modules are therefore also constant.

Figure 9(e) and 9(f) show, in linear and logarithmic scale, the mean number of minimal diagnoses per agent. For all systems with more than one agent, both the size and the number of minimal diagnoses are reduced for the set of minimal condensed diagnoses compared to the set of minimal global diagnoses. The reduction in mean number of diagnoses is for example 70 % for the system with two agents, increasing to 95 % reduction for the system with four agents.

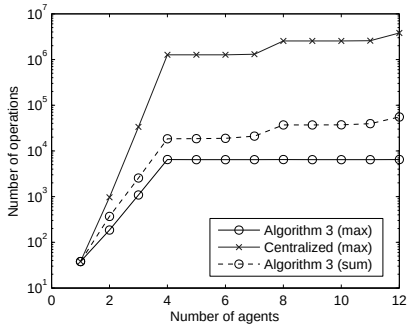
# 9 REMOVING THE SIGNAL ASSUMPTION

When designing the algorithms in Section 5, it is assumed that Assumption 1 is fulfilled. The assumption stated that, the dependency for a signal, which is an output from an agent, is limited to the set of private components in the agent, and that all dependencies are disjoint. The assumption is taken to make the algorithms more readable, and since the assumption by construction is fulfilled in many industrial applications. However, systems exist for which the assumption is not fulfilled, and therefore, this section will show how the algorithms can be adapted such that the assumption can be removed. Section 9.1 will show how the assumption of disjoint signal dependencies can be removed and Section 9.2 will show how the dependencies can be extended to the set of common components.

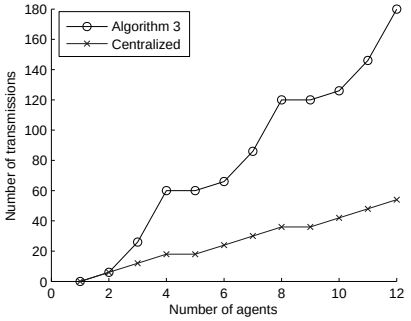
In an application, it should first be checked if Assumption 1 is fulfilled by construction. If this is not the case, then it should be consid-



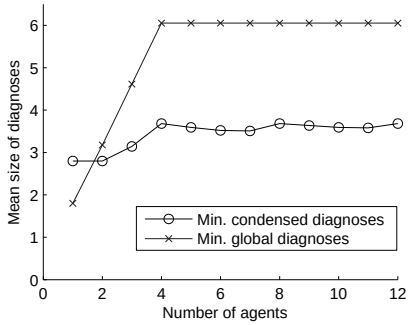

(a) Mean maximum number of operations in any agent. The zoom is from the bottom left corner.



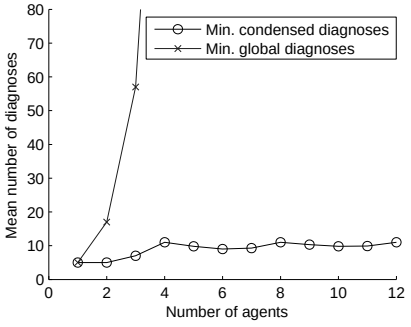

(b) Mean maximum number of operations in any agent, logarithmic scale.



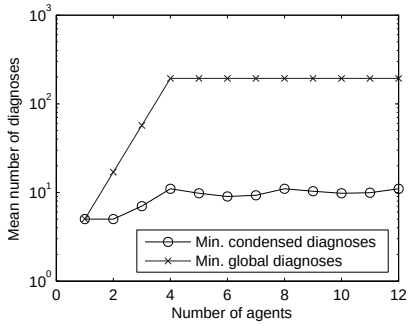

(c) Transmissions.




(d) Mean size of diagnoses.




(e) Mean number of diagnoses in each agent.




(f) Mean number of diagnoses in each agent, logarithmic scale.

FIGURE 9: The mean number of operations, transmissions, minimal diagnoses in each agent, and mean size of the minimal diagnoses.

ered that maybe the system should be redesigned such that Assumption 1 is fulfilled by construction. The reason for this is that, without Assumption 1, it is not possible to assume independency of signals, making it difficult to use one of them to supervise the other, and similar for the common components.

## 9.1 Non Disjoint Signal Dependencies

This section will extend Algorithm 3 such that the assumption that the dependencies are disjoint can be removed. However, first a discussion about the effect that non-disjoint dependencies have on the algorithm.

### Discussion

Consider for example two signals $s_1$ and $s_2$ that depend on the component $A$ and assume that this component has been found to behave abnormally. In this case, both signals should by definition be included in the corresponding condensed diagnosis. Similarly, even though an agent has only detected that one of the signals are behaving abnormally, both of the signals should be included in the corresponding condensed diagnosis. If for example signal $s_1$ has been found to behave abnormally, this means that component $A$ is included in a minimal global diagnosis, which means that not only $s_1$ but also $s_2$ should be included in the corresponding condensed diagnosis. This type of dependencies has to be managed in an algorithm that aims at computing the set of minimal condensed diagnoses, such as the extended algorithm that will be designed below.

However, to keep the algorithm more readable, it will here instead be assumed that if a signal is included in a minimal local diagnosis then each signal, which depends on any of the components that the first signal depends on, is also included in the minimal local diagnosis. If for example signal $s_1$ is included in a minimal local diagnosis, then all local diagnoses including $s_1$ should also include the signal $s_2$ since they both depend on component $A$. Considering responded diagnostic tests, this assumption means that, if a signal is included in a generated conflict, then for each signal, which depends on any of the components that the first signal depends on, an additional conflict is generated that equals the original conflict but where the first signal has been replaced with one of the other signals. If for example a diagnostic test that supervises signal $s_1$ has responded, then the two conflicts $\{s_1\}$ and $\{s_2\}$ should be generated. From this follows then that the local diagnosis $\{s_1, s_2\}$ is computed that fulfills the assumption that signals depending on the same components should all be included in the local diagnosis if any of the signals is included.

Further, considering industrial applications, the information about such non-disjoint dependencies should naturally be included in the description of the signals characteristics.

### Extending Algorithm 1–3 to Non-Disjoint Dependencies

The following modifications of the algorithms in Section 5 will extend Algorithm 1–3 such that signals with non-disjoint dependencies can be included in the system.

In Algorithm 2, remove row 12, and replace `CrossProduct` in row 6 with `MinimalHittingSets`. The input to the minimal hitting set operation is a set of dependencies. If for example a signal $s_1$ has a dependency $\texttt{Dep}(s_2) = \{A, B\}$ and another signal $s_2$ has a dependency $\texttt{Dep}(s_2) = \{A, C\}$, then the set of signals $\{s_1, s_2\}$ is replaced with $\texttt{MinimalHittingSets}(\{\texttt{Dep}(s_1), \texttt{Dep}(s_2)\}) = \{\{A\}, \{B, C\}\}$.

A case that could now arise is that if several signals in a condensed diagnoses depend on the same component, then the cardinality of the condensed diagnosis will be to high. Consider for example the signals $s_1$ and $s_2$ that both depend on the component $A$. The cardinality of the condensed diagnosis $\langle\{s_1, s_2\}, 0\rangle$ is two while the cardinality of the corresponding global diagnosis $\{A\}$ is only one. To correct this, a compensation should be performed such that the condensed diagnosis becomes $\langle\{s_1, s_2\}, -1\rangle$ that has a correct cardinality.

The compensations are performed by adding the following rows to Algorithm 2.

12: **for each** $\langle D, k\rangle \in \mathbb{D}_c^{A_i}$ where $D = \cup_j D_j$ and $D_j = \bar{P}^j \cup G^j \cup \bar{\Gamma}^j \cup \Omega^j$ **do**
13: $\quad k := k - \texttt{Compensate}(\cup_j \bar{\Gamma}^j \setminus \cup_j \Omega^j)$
14: **end for**
15: $\mathbb{D}_c^{A_i} := \texttt{minimal\_condensed\_diagnoses}(\mathbb{D}_c^{A_i})$

When computing the compensation, the set of signals partitioned into the sets $\bar{\Gamma}^j$ and $\Omega^j$ is used, where these sets are computed in row 4 in the algorithm. The compensation function is

$$\texttt{Compensate}(\Gamma) = |\Gamma| - \min_{H \in \texttt{MinimalHittingSets}(\cup_{s\in\Gamma}\{\texttt{Dep}(s)\})} |H|$$

for a set of signals $\Gamma$. The compensation is the number of signals minus the minimal number of components that could have caused the signals in $\Gamma$ to be abnormal. Notice the similarity with row 4–5 in Algorithm 1 where the compensation is performed for the signals that are outputs from the transmitting agent. In the algorithm, the input to $\texttt{Compensate}(\cdot)$ is the signals that are outputs from agent $A_j$ except those that have already been compensated for in Algorithm 1 that are signals in the set $\Omega^j$. The compensation might seem cumbersome, however, the compensation is in many applications zero for most combinations of signals since the dependencies of the signals are

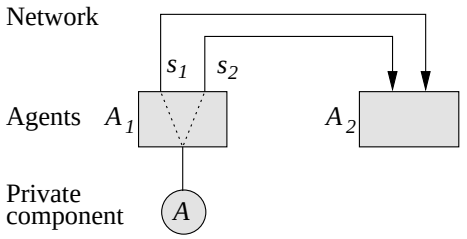



FIGURE 10: An example of a system consisting of two agents, one component, and two signals.

often disjoint. If the signals are disjoint then the set of signals $\Gamma$ have the same cardinality as the set $H$ in $\texttt{Compensate}(\Gamma)$, since there is one component in $H$ for each signal in $\Gamma$. However, for those signals that are not disjoint, the compensation can be computed off-line and saved for use in the agents when Algorithm 2 is evaluated. Consider for example the signals $s_1$ and $s_2$ described above, the compensation is always $-1$ when both of these signals are included in a diagnosis.

The following example will illustrate two different cases that can arise when using the algorithm.

EXAMPLE 5: Consider an agent $A_1$ with private component $A$ and two output signals $s_1$ and $s_2$ that both depend on the component $A$, see Figure 10. Assume that agent $A_2$ has detected that the two signals are behaving abnormally and has computed the minimal local diagnosis $\{s_1, s_2\}$. Further, assume that agent $A_1$ only has the empty minimal local diagnosis. Using Algorithm 3, the minimal condensed diagnosis $\langle\{s_1, s_2\}, 0\rangle$ in $A_1$ is calculated before the compensation is performed. The input to $\texttt{Compensate}(\cdot)$ is the set $\bar{\Gamma}^1 \cup \bar{\Gamma}^2 \setminus \Omega^2 = \{s_1, s_2\}$ since $\bar{\Gamma}^2 = \{s_1, s_2\}$ and $\bar{\Gamma}^1 = \Omega^2 = \emptyset$. The compensation is therefore $\texttt{Compensate}(\{s_1, s_2\}) = |\{s_1, s_2\}| - |\{A\}| = 1$. The result of the compensation is the minimal condensed diagnosis $\langle\{s_1, s_2\}, -1\rangle$ that represent the minimal global diagnosis $\{A\}$. The compensations are here performed in the receiving agent.

Assume now instead that $A_1$ has detected that its private component $A$ is behaving abnormally and has computed the local diagnosis $\{A\}$. Further, assume that agent $A_2$ only has the empty minimal local diagnosis. Agent $A_1$ replaces the private component with the signals and transmits the set $\{\langle\{s_1, s_2\}, -1\rangle\}$. In $A_2$, the minimal condensed diagnosis $\{\langle\{s_1, s_2\}, -1\rangle\}$ is computed. In this case, the input to $\texttt{Compensate}(\cdot)$ is $\bar{\Gamma}^1 \cup \bar{\Gamma}^2 \setminus \Omega^1 = \emptyset$, where $\bar{\Gamma}^1 = \bar{\Gamma}^2 = \emptyset$ and $\Omega^1 = \{s_1, s_2\}$, and the compensation is therefore 0. The compensation had in this case already been done in the transmitting agent, while in the first case, the compensation had to be performed in the receiver. $\diamond$

## 9.2 Signals Depending on Common Components

This section will give an extension of Algorithm 3 such that dependencies can include common components, i.e. $\texttt{Dep}(s) \subseteq \mathcal{P}^{A_i} \cup \mathcal{G}$ for a signal $s$ from agent $A_i$. This is in contrast to Assumption 1 that required that $\texttt{Dep}(s) \subseteq \mathcal{P}^{A_i}$. If wanted, the extended versions of Algorithm 1–3, described in Section 9.1, can be used.

The problem that arise when a dependency includes common components is that both the common component and the signals depending on the common component must be included in the corresponding condensed diagnosis. Assume for example that a signal $s$ only depends on the common component $G$, i.e. $\texttt{Dep}(s) = \{G\}$. If $G$ is included in a minimal local diagnosis $\{G\}$, then both $G$ and $s$ should by the definition of a condensed diagnosis be included in the condensed diagnosis $\langle\{G, s\}, -1\rangle$. However, since Algorithm 3 does not deal with the case of signal dependency on common components, the designed algorithm will give the minimal condensed diagnosis $\langle\{G\}, \emptyset\rangle$, which is incorrect. The reason why both $G$ and $s$ should be included is that otherwise, if for example only $G$ is included in a diagnosis in an agent, then the agent might incorrectly presume that component $s$, which is also used by the agent, is normal. The knowledge that if $G$ is abnormal then $s$ is also abnormal should be available to the agent, which is fulfilled if the condensed diagnoses are used.

To be able to deal with the dependency on common components, it is here suggested that a post-update of the sets of minimal condensed diagnoses is performed. Meaning that after the Algorithm 3 has been evaluated, each minimal condensed diagnosis is updated such that it becomes a true minimal condensed diagnosis. Given the set of minimal condensed diagnoses $\mathbb{D}_c^{A_i}$ in agent $A_i$, add the following rows to Algorithm 3.

10: $\bar{\mathbb{D}}_c^{A_i} := \emptyset$
11: **for each** $\langle P \cup G \cup \Gamma, k\rangle \in \mathbb{D}_c^{A_i}$ **do**
12: $\quad\check{\Gamma} := \{s : \texttt{Dep}(s) \cap G \neq \emptyset\}$
13: $\quad$**for each** $\check{G} \in \texttt{MinimalHittingSets}(\{dep(i) \cap \mathcal{G} : s \in \check{\Gamma}\})$ **do**
14: $\quad\quad\check{k} := |\Gamma| - |\Gamma \cup \check{\Gamma}| + |G| - |G \cup \check{G}|$
15: $\quad\quad\bar{\mathbb{D}}_c^{A_i} := \bar{\mathbb{D}}_c^{A_i} \cup \{\langle P \cup G \cup \check{G} \cup \Gamma \cup \check{\Gamma}, k + \check{k}\rangle\}$
16: $\quad$**end for**
17: **end for**
18: $\mathbb{D}_c^{A_i} := \bar{\mathbb{D}}_c^{A_i}$

For each condensed diagnosis $\langle P \cup G \cup \Gamma, k\rangle$ in $\mathbb{D}_c^{A_i}$, this update adds signals $\check{\Gamma}$ whose dependency includes any component in $G$. For each minimal combination $\check{G}$ of common components depending on the signals in $\check{\Gamma}$, an updated condensed diagnosis $\langle P \cup G \cup \check{G} \cup \Gamma \cup \check{\Gamma}, k + \check{k}\rangle$ is created, where $\check{k}$ compensates for the new set of signals $\check{\Gamma}$ and the new set of common components $\check{G}$ that has been added to the diagno-

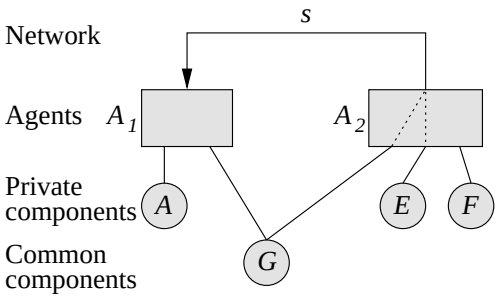



FIGURE 11: An example of a two agent system where the signal depends on a common component.

sis. The compensation is done performed such that the cardinality is consistent with the corresponding global diagnosis.

The following example will illustrate the extended algorithm.

EXAMPLE 6: Consider the system shown in Figure 11. A signal $s$ exists whose dependency is $\mathtt{Dep}(s) = \{G, E\}$, represented by the dotted line. The sets of private components are $\mathcal{P}^{A_1} = \{A\}$ and $\mathcal{P}^{A_2} = \{E, F\}$. The set of common components is $\mathcal{G} = \{G\}$.

Assume that the local diagnosis in agent $A_2$ is $\{G\}$ and agent $A_1$ has only the empty diagnosis. The diagnosis in $A_2$ is transmitted to agent $A_1$ as $\{\langle\{G\}, 0\rangle\}$, where it is received and merged to the minimal condensed diagnosis $\{\langle\{G\}, 0\rangle\}$. However, the minimal global diagnosis is $\{G\}$ and this shows that the minimal condensed diagnosis should be $\langle\{G, s\}, -1\rangle$. Using the update given above, the minimal condensed diagnosis is updated to the minimal condensed diagnosis $\{\langle\{s, G\}, -1\rangle\}$ that is consistent with the minimal global diagnosis.

Assume instead that agent $A_2$ has the diagnosis $\{E\}$ and agent $A_1$ the diagnosis $\{G\}$. The tuple $\langle\{s\}, 0\rangle$ is transmitted to agent $A_1$ where the condensed diagnosis $\langle\{G, s\}, 0\rangle$ is computed. The update does not change the condensed diagnosis since $\check{\Gamma} = \Gamma$ and $\check{G} = G$. The global diagnosis is in this case $\{G, E\}$ that shows that the condensed diagnosis is correct. ◇

This section has given extensions of the algorithms designed in Section 5 such that both parts of Assumption 1 could be removed. The removal of the assumption removes limitations on the signals dependencies but increases the complexity of the algorithms. Due to the increased complexity, if a system does not fulfilled Assumption 1 then it should first be considered if the system can be changed such that the assumption is fulfilled. If it is not possible or desirable to change the system, the extended algorithms designed in this section can be used when computing the sets of minimal condensed diagnoses.

# 10 CONCLUSIONS

A distributed algorithm has been designed that computes a set of minimal or minimal cardinality condensed diagnoses in each agent, where only components affecting the behavior of the agent are included. The sets of minimal or minimal cardinality condensed diagnoses are computed directly from the sets of minimal local diagnoses and the algorithm therefore do not require the computation of the set of minimal or minimal cardinality global diagnoses respectively. Due to the removal of unaffecting components, the number of minimal condensed diagnoses in each agent is reduced compared to the set of minimal global diagnoses. For the automotive application including three agents, 85 components, and over 100 diagnostic tests, the number of diagnoses is for example reduced with up to 90 %. Due to this reduction, the minimal condensed diagnoses are suitable to be used when for example repair of an agent should be performed since fewer diagnoses have to be checked. Using the designed algorithm, both the computational load and the memory usage are reduced compared to a centralized algorithm that computes the set of minimal global diagnoses. The automotive application showed a reduction of up to 85 % in both maximum processor load and memory usage when the sets of minimal condensed diagnoses were computed.

# *Paper III*

# Determining the Fault Status of a Component and its Readiness, with a Distributed Automotive Application[1]

**Jonas Biteus**[⋆], **Mattias Nyberg**[†], **Erik Frisk**[⋆], **and Jan Åslund**[⋆]

[⋆] *Dep. of Electrical Engineering, Linköpings universitet,*
SE-*581 83 Linköping, Sweden.*
{biteus,frisk,jaasl}@isy.liu.se.
[†] *Power-train division, Scania,* SE-*151 87 Södertälje, Sweden.*
mattias.nyberg@scania.com

## Abstract

In automotive vehicles using only single component tests, the fault status of a component is ready if a test only supervising the component has been evaluated. However, if plausibility tests that supervise multiple components are used, then a component can be ready before all tests supervising the component have been evaluated. Based on test results, this paper contributes with conditions on when a component is ready, and a strategy that decides which test whose evaluation will give the most ready components. The number of tests that have to be evaluated to gain readiness can thereby be minimized. The conditions on readiness are given for both centralized and distributed systems and are applied to the distributed diagnostic system in an automotive vehicle.

[1] An earlier and shorter version of this paper has been presented in (Biteus et al., 2006b).

# 1 INTRODUCTION

Fault diagnosis is important in many applications and has become even more so with the increase of on-board computing power in technical processes, see for example (Isermann, 2005). This work is motivated by the diagnostic systems, i.e. the system responsible for fault diagnosis and supervision, used in automotive vehicles (Gertler, 1998; Hristu-Varsakelis and Levine, 2005) and in particular that used in a Scania heavy duty vehicle. These systems typically store a diagnostic trouble code (DTC) when a component is found to be faulty (SAE, 2003; ISO, 1999). In the first generations of diagnostic systems, each diagnostic test checked exactly one component for abnormal behavior. Therefore, the DTCs could be used to exactly state the fault status of a component, which is faulty if a test supervising the component has responded and normal otherwise.

Due to higher demands on fault diagnosis, such as reduced emission levels (EU, 2005), more components must today be supervised for abnormal behavior compared to the first generations of diagnostic systems. Since the number of sensors in the system is limited, the industry has been forced to introduce diagnostic tests that check the correct behavior of several components at the same time, often denoted plausibility tests. The plausibility tests are for example based on analytical redundancy relations (ARR) (Gertler, 1998; Isermann, 2001; Nyberg, 1999b). The plausibility tests come into conflict with the framework based on single component tests that has previously been used in automotive vehicles. With the addition of plausibility tests, a component can now be only suspected to behave abnormally and its fault status is therefore suspected.

In addition to the fault status, if fault diagnosis is performed using precomputed diagnostic tests then there is an advantage if the repair technician or the control system can get an indication on when the evaluation of additional diagnostic tests can not change the fault status of a component, and the fault status is in this case said to be ready. The evaluation of all tests give readiness for all components, however, due to for example limited processing power, it is in automotive applications not always possible to evaluate all tests. Therefore, the cost in evaluating a test has to be weighted against improvement in fault diagnosis. If for example the evaluation of additional diagnostic tests will facilitate the repair by improving the fault statuses, it is an advantage to evaluate these tests before the repair is started. In automotive applications, readiness codes are used to state if all tests supervising a component or a subsystem have been evaluated or not (ISO, 1999). The difference between readiness in (ISO, 1999) and this paper is that a component might here be ready even though not all diagnostic tests have been evaluated.

This paper contributes with an algorithm, denoted the FSR (fault status and readiness) algorithm, which computes the fault status and readiness for all components. The FSR algorithm is here applied to the diagnostic system used in a heavy duty vehicle from Scania.

The notation of fault status and readiness are defined with respect to a centralized system. However, many applications, and especially those used in the automotive industry, include multiple electronic control units (ECUs), generally denoted agents, which includes diagnostic systems (Leen and Heffernan, 2002; Navet et al., 2005; Hristu-Varsakelis and Levine, 2005; Biteus, 2005). In these distributed systems, diagnostic tests might exist in one agent that supervise components that belong to another agent. Therefore, an additional contribution of this paper is that the notations of fault status and readiness are extended to distributed systems. This extension render it possible use the fault status computed locally, to state the fault status applicable for the complete distributed system. The application on the heavy duty vehicle is extended to the distributed case.

One of the objectives for a diagnostic system is to achieve readiness for the fault status of a component. Therefore, conditions are stated and an algorithm is designed that computes which diagnostic tests that should be evaluated to achieve readiness for a specific component. Further, given a set of such diagnostic tests, a strategy is designed that schedule the evaluation of the tests such that readiness is achieved using as few evaluations of tests as possible.

In the AI field (Reiter, 1987; Dressler and Struss, 1996), the dominant methodology for fault diagnosis has been so called consistency based diagnosis, on which this paper is based. The methodology of consistency based diagnosis has strong relationships with the methods for fault diagnosis used in engineering disciplines (Cordier et al., 2004), such as control theory and statistical decision making (Gertler, 1998; Gertler et al., 1995; Basseville and Nikiforov, 1993). Within this methodology, a diagnosis points at a set of components whose abnormal behavior could explain why a system does not function as intended and are primarily used for repair and fault tolerant control (Shin and Belcastro, 2006). The fault status differs from diagnoses in that the fault statuses give the components that certainly behave abnormally, which that might behave abnormally, and those that are not behaving abnormally, while each diagnosis points at a set of components that might behave abnormally. Further, it is in some applications intractable to compute the diagnoses since the complexity increases exponentially with the number of tests, this is in contrast to the fault status that has only a linearly increasing complexity and is therefore in practice always tractable.

# 2 BACKGROUND TO FAULT DIAGNOSIS

In this section the framework in consistency based diagnosis will be introduced. This framework will be used in the rest of this paper.

A system consists of a set of components $\mathcal{C}$ that should be supervised by the diagnostic system. A component is something that can be diagnosed, such as sensors, actuators, cables, and pipes. Here, only the abnormal and the not abnormal mode is considered, where the abnormal mode does not have a model. Further, the set notation used in for example GDE is employed (Kleer and Williams, 1987).

A diagnosis $D \subseteq \mathcal{C}$ is a set of components, such that the abnormal behavior of the components in D, the normal behavior of the remaining components, the system description, and the observations are consistent. All supersets of a diagnosis D are also diagnoses since only the abnormal mode, without a model, and the not abnormal mode are considered. Further, a diagnosis D is a minimal diagnosis if there is no proper subset $D' \subset D$ where $D'$ is a diagnosis (Kleer et al., 1992).

A conflict is a set of components $\pi \subseteq \mathcal{C}$, such that the normal behavior of the components, the system description, and the observations are inconsistent. As with diagnoses, a conflict $\pi$ is a minimal conflict if there is no proper subset $\pi' \subset \pi$ where $\pi'$ is a conflict. If a single-component diagnostic test responds, i.e. if it detects that the component c that it supervises is behaving abnormally, then a conflict $\pi = \{c\}$ is generated. Further, if a plausibility test responds then a conflict $\pi \subseteq \mathcal{C}$ is generated. From the definitions follow that a set $D \subseteq \mathcal{C}$ is a diagnosis if and only if it has a nonempty intersection with every conflict. A consequence of this is that the set of minimal diagnoses is the set of minimal hitting sets for the set of minimal conflicts (Kleer et al., 1992).

# 3 FAULT STATUS AND READINESS

This section will focus on systems with one agent, while Section 4 will extend the results to distributed systems consisting of multiple agents.

## *3.1 Component Fault Status: Faulty, Suspected, and Normal*

It was in Section 1 stated that the fault status of each component is wanted, where the fault status is either faulty, suspected, or normal. Here, the fault status will be defined with respect to the minimal conflicts since this will clearly show how the fault status relates to how DTCs are set in automotive industry. However, equivalent results would follow if the fault status were defined with respect to the set of minimal diagnoses, which is shown in Proposition 1.

DEFINITION 1: *Let* $\Pi$ *be the set of minimal conflicts. The fault status of component* $c$ *is* faulty *if and only if*

$$\exists \pi \in \Pi : \pi = \{c\}. \tag{1}$$

*The fault status of component* $c$ *is* suspected *if and only if*

$$\exists \pi \in \Pi : c \in \pi \wedge |\pi| > 1. \tag{2}$$

*The fault status of component* $c$ *is* normal *if and only if*

$$\forall \pi \in \Pi : c \notin \pi. \tag{3}$$

The possible values of the fault status for a component are exhaustive, i.e. a component is either faulty, normal, or suspected. It follows from the definition that when only single-component diagnostic tests are used, then it is only possible for the fault status of a component to be faulty or normal. This coincides with how DTCs were set in automotive vehicles before plausibility tests were introduced. The definition of normal fault status is reasonable from an engineering perspective, since if there is no indication that a component is behaving abnormally then there is no reason to repair the component and its fault status can therefore be denoted normal.

EXAMPLE 1: Consider a system consisting of the components $A$, $B$, $C$, $D$, and $E$. Let diagnostic tests exist such that the possible conflicts are $\{A\}$, $\{B, C\}$, $\{C\}$, and $\{B, D\}$. If the present conflicts are $\{A\}$ and $\{B, C\}$ then it follows from Definition 1 that the fault status of component $A$ is faulty, and the fault status of $B$ and $C$ are suspected. Further, the fault status of the rest of components, i.e. $D$ and $E$, are normal. $\diamond$

Since the diagnoses commonly are used within the AI field, the relation between diagnoses and fault status is here given.

PROPOSITION 1: *Let* $\mathcal{D}$ *be the set of minimal diagnoses. The fault status of component* $c$ *is* faulty *if and only if*

$$\forall D \in \mathcal{D} : c \in D. \tag{4}$$

*The fault status of component* $c$ *is* suspected *if and only if*

$$(\exists D \in \mathcal{D} : c \in D) \wedge (\exists D \in \mathcal{D} : c \notin D). \tag{5}$$

*The fault status of component* $c$ *is* normal *if and only if*

$$\forall D \in \mathcal{D} : c \notin D. \tag{6}$$

*Proof.* The proposition is proved using the fact that the set $\mathcal{D}$ is the set of minimal hitting sets w.r.t. the set of minimal conflicts $\Pi$. Faulty:

Component $c \in D$ for all minimal diagnoses $D \in \mathcal{D}$ if and only if a conflict $\pi = \{c\}$ exists. By Definition 1, it then follows that (4) holds if and only if c is faulty. Suspected: A diagnosis $D \in \mathcal{D}$ exists where $c \in D$ if and only if a conflict $\pi$ exists where $c \in \pi$. Further, $D \in \mathcal{D}$ exists where $c \notin D$ if and only if a conflict $\pi = \{c\}$ does not exist. By Definition 1, it then follows that (5) holds if and only if c is suspected. Normal: By Definition 1, it follows directly that (6) holds if and only if c is normal. □

Proposition 1 shows that if a component is not included in the minimal diagnoses, then its fault status is normal. Similar to Definition 1, this is reasonable since there is no indication in the minimal diagnoses that the component is behaving abnormally.

EXAMPLE 2: Continuation of Example 1. The set of minimal diagnoses for the present conflicts $\{A\}$ and $\{B, C\}$ is the set

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

It follows from Proposition 1 that the fault status of component A is faulty, the fault status of B and C are suspected, and the fault status of D and E are normal. As should be, this coincides with the fault statuses computed directly using Definition 1. ◇

## 3.2 *The Readiness of the Fault Status*

In this section, the readiness of the fault status will be defined and conditions useful to compute the readiness from the conflicts will be given.

DEFINITION 2: *The fault status of component* c *is* ready *if and only if the fault status of* c*, considering the present minimal conflicts* $\Pi$*, is unchanged for all possible future sets of minimal conflicts* $\bar{\Pi}$.

The set $\Pi$ is the set of conflicts resulting from the evaluated and responded diagnostic tests. The non evaluated diagnostic tests and the evaluated but non responded tests could in the future give the set of conflicts $\Pi^f$. Therefore, the set of future minimal conflicts $\bar{\Pi}$ mentioned in Definition 2 is some set of minimal conflicts from the set $\Pi \cup \Pi^f$.

For a component whose fault status is faulty, the following simple relation between the readiness and the fault status holds.

PROPOSITION 2: *Let the fault status of component* c *be* faulty*, then the fault status of* c *is ready.*

*Proof.* A conflict $\pi = \{c\}$ exists if and only if the fault status of c is faulty, Definition 1, and since $\pi$ is always a minimal conflict, the fault

status is always faulty, and from Definition 2 it then follows that the fault status of c is ready. □

The proposition shows that the definition of readiness follows the intuitive meaning of faulty. If a component has been found to certainly behave abnormally, then it can in the future not be found to not behave abnormally.

The following two propositions state the relation between readiness and conflicts when the fault status is suspected or normal.

PROPOSITION 3: *Let* $\Pi$ *be the set of present minimal conflicts and let* $\Pi^f$ *be the set of all possible future conflicts. Let the fault status of component* c *be* suspected*, then the fault status of* c *is ready if and only if*

$$(\nexists \pi^f \in \Pi^f : \pi^f = \{c\}) \wedge \tag{7a}$$

$$\big(\exists \pi \in \Pi : c \in \pi \wedge (\nexists \pi^f \in \Pi^f : c \notin \pi^f \wedge \pi^f \subset \pi)\big). \tag{7b}$$

*Proof.* Component c is suspected if and only if (2) holds. For all future conflicts, the first and second part of (2) hold if and only if (7b) and (7a) hold respectively. The minimal diagnoses are exactly defined by the minimal conflicts, and by Definition 2, it therefore follows that c is ready if and only if (7) holds. □

The meaning of (7b) is as follows: If a possible future conflict is a subset of the present conflict, which made the fault status of c suspected, then the present conflict is in the future non-minimal and is therefore removed. Further, if one such future conflict exists for each conflict that made c suspected, then the fault status of c would no longer be suspected. This type of subset condition will be found in several of the propositions stated in this paper.

It is straightforward to construct an algorithm that test an equation such as (7) by testing all conflicts. However, such a direct implementation will not be very efficient due to the testing of all conflicts, therefore, the FSR algorithm designed in Section 5 will be based on a graph-representation of the conflicts that will make it possible to compute the fault statuses and readiness without testing all conflicts.

EXAMPLE 3: Consider Example 1 where the present conflicts are $\{A\}$ and $\{B, C\}$, and the possible future conflicts are $\{C\}$ and $\{B, D\}$. The fault status of components B and C are suspected. Using Proposition 3, it can be found that component B is not ready since for the possible future conflict $\pi^f = \{C\}$ and for the present conflict $\pi = \{B, C\}$, the fault status of B becomes normal and the condition in (7b) is therefore not fulfilled. The fault status of C is also not ready since for the possible future conflict $\pi = \{C\}$, the fault status of C becomes faulty and the condition in (7a) is therefore not fulfilled. ◇

PROPOSITION 4: *Let* $\Pi$ *be the set of present minimal conflicts, and let* $\Pi^f$ *be the set of all possible future conflicts. Let the fault status of component* $c$ *be* normal*, then the fault status of* $c$ *is ready if and only if*

$$\nexists \pi^f \in \Pi^f : c \in \pi^f \wedge (\forall \pi \in \Pi : \pi \not\subset \pi^f). \tag{8}$$

*Proof.* The fault status of $c$ is normal for all possible future diagnoses if and only if $c \notin \pi^f$ for each conflict $\pi^f \in \Pi^f$ that is non-minimal considering $\Pi$, Definition 1. The fault status is therefore normal for all possible future diagnoses if and only if (8) holds. From Definition 2 follows now that $c$ is ready if and only if (8) holds. □

EXAMPLE 4: Consider once again Example 1. The fault status of $D$ and $E$ are normal. From Proposition 4 follows that the fault status of $D$ is not ready since a possible future conflict $\pi^f = \{B, D\}$ exist that does not fulfill condition (8). The fault status of $E$ on the other hand is ready since any possible future conflict $\pi^f$ does not exist where $E \in \pi^f$, and thereby fulfilling (8). ◇

In summary: The conditions in Proposition 2, 3, and 4 can be used to decide if the fault status of a component is ready. The conditions in these propositions relates the readiness to the conflicts and are therefore suitable to use when constructing an algorithm that computes the readiness.

# 4 DISTRIBUTED SYSTEMS

The fault status and the readiness are here extended to distributed systems such that it is possible to use the fault status and readiness computed locally in each agent to state the fault status and readiness for the complete system. First, distributed systems will be exemplified and a framework for distributed systems will be designed. The fault status and its readiness will then be extended to distributed systems.

## *4.1 An Example of a Distributed System*

Figure 1 shows a configuration of the distributed system used in current Scania heavy-duty vehicles. It includes three separate CAN (controller area network) buses, which are connected to the coordinator ECU. Each ECU is further connected to sensors and actuators, and both sensor values and control signals can be shared with the other ECUs over the network. There are between 20 and 30 ECUs in the system, depending on the configuration of the truck, and between 4 and 110 components are supervised by the diagnostic system in each ECU. The diagnostic tests in some of the ECUs supervise components physically

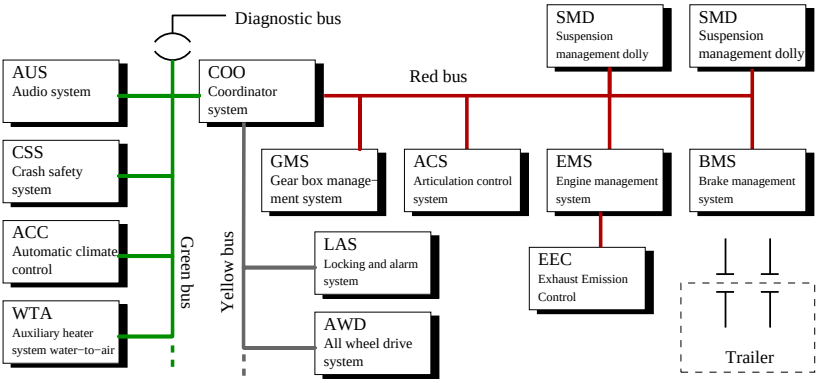



FIGURE 1: A part of the distributed system in Scania vehicles.

connected to other ECUs trough the sharing of for example sensor values. A consequence of this is that the fault status of a component can locally be stated by several ECUs independent of the other ECUs.

## 4.2 *Framework for Distributed Fault Diagnosis*

Based on the example system given above, a framework suitable for stating the fault status and readiness in a distributed system has been developed.

A distributed system consists of a set of components $\mathcal{C}$, which should be supervised by the diagnostic systems implemented in a set of agents $\mathcal{A}$. A local diagnosis is determined by the conflicts in a single agent, while a global diagnosis is determined by the conflicts in all agents. In more detail, let $\Pi^A$ be the set of minimal conflicts detected in agent $A \in \mathcal{A}$, and let $\mathbb{D}^A$ be the set of local minimal diagnoses determined by the set of minimal conflicts $\Pi^A$. The set of minimal global diagnoses $\mathcal{D}$ is determined by the set of minimal conflicts in the set $\cup_{A\in\mathcal{A}}\Pi^A$.

## 4.3 *Faulty, Suspected, and Normal Fault Status*

The fault status of a component can, in a distributed system, be divided into two different levels, the global and the local fault status.

DEFINITION 3: *The* global fault status *(*GFS*) of component* $c$ *is faulty, suspected, or normal if it is faulty, normal, or suspected, respectively, with respect to the set of minimal conflicts in the set* $\cup_{A\in\mathcal{A}}\Pi^A$.

DEFINITION 4: *The* local fault status *(*LFS*) of component* $c$ *is faulty, normal, or suspected, for agent* $A$ *if it is faulty, normal, or suspected, respectively with respect to the set of minimal conflicts* $\Pi^A$ *in agent* $A$.

The GFS can be computed from the set of all minimal conflicts in all agents using Definition 1. However, if the LFSes have been computed for all agents, it is possible to increase the efficiency in the computation of the GFS by using the LFSes. Such relations are here given for the three cases of faulty, suspected, and normal GFS.

The GFS has a simple relation to the LFS in some agent where the LFS is faulty.

PROPOSITION 5: *The* GFS *of component* c *is* faulty *if and only if the* LFS *is faulty for some agent.*

*Proof.* The GFS of component c is faulty if and only if (1) holds considering all conflicts in all agents, and this is the case if and only if the LFS of c is faulty for some agent. □

The proposition shows that the definition of globally faulty follows the intuitive meaning of faulty. If the LFS of a component is faulty, then its GFS must also be faulty. The relation between the GFS suspected and normal, and the LFS is given by the following propositions.

PROPOSITION 6: *The* GFS *of component* c *is* suspected *if and only if*

$$(\nexists A \in \mathcal{A} : (\textit{the LFS of } c \textit{ in } A \textit{ is faulty})) \wedge \tag{9a}$$

$$\big(\exists A \in \mathcal{A} : (\textit{the LFS of } c \textit{ in } A \textit{ is suspected}) \wedge \tag{9b}$$

$$(\exists \pi \in \Pi^A : c \in \pi \wedge (\nexists \tilde{\pi} \in \bigcup_{\tilde{A} \in \mathcal{A} \setminus A} \Pi^{\tilde{A}} : \tilde{\pi} \subset \pi))\big). \tag{9c}$$

*Proof.* The GFS of component c is suspected if and only if (2) holds w.r.t. the minimal set of all conflicts in all agents. By Proposition 5, second part of (2) holds if and only if (9a) hold and first part of (2) holds if and only if (9b) and (9c) hold. □

PROPOSITION 7: *The* GFS *of component* c *is* normal *if and only if*

$$\forall A \in \mathcal{A} : (\textit{the LFS of } c \textit{ in } A \textit{ is normal}) \vee \tag{10a}$$

$$\big((\textit{the LFS of } c \textit{ in } A \textit{ is suspected}) \wedge \tag{10b}$$

$$(\forall \pi \in \{\pi \in \Pi^A : c \in \pi\} : (\exists \tilde{\pi} \in \bigcup_{\tilde{A} \in \mathcal{A} \setminus A} \Pi^{\tilde{A}} : \tilde{\pi} \subset \pi))\big). \tag{10c}$$

*Proof.* The GFS of component c is normal if and only if (3) holds. Equation (3) holds if and only if for each agent, the LFS is normal (10a), or it is suspected (10b) and the conflict including c is non-minimal w.r.t. the minimal set of conflicts in all agents if and only if (10c) holds. The GFS of c is therefore normal if and only if (10) holds. □

EXAMPLE 5: A system consists of two agents $A_1$ and $A_2$ that have computed the sets of minimal conflicts $\Pi^{A_1} = \{\{A, B\}\}$ and $\Pi^{A_2} = \{\{A\}, \{C, D\}\}$ respectively. The LFSes of components A and B are suspected in agent $A_1$, while in $A_2$ the LFS of component A is faulty and the LFSes of C and D is suspected.

Using Proposition 5 the GFS of A can be found to be faulty since an LFS exist where A is faulty. Further, the GFS of B is normal since the LFS of B is normal in $A_2$, i.e. (10a), and in $A_1$ it is both suspected, i.e. (10b), and a conflict $\{A\}$ exist in $A_1$ for which $\{A\} \subset \{A, B\}$, i.e. (10c). Finally, the GFSes of C and D are suspected since they are suspected in $A_2$ and the conflict $\{A, B\} \not\subset \{C, D\}$. To conclude, the GFS of A is faulty, C and D are suspected, and B is normal.

To verify the result, the set of minimal global diagnoses $\mathcal{D} = \{\{A, C\}, \{A, D\}\}$ has been computed. It can be seen that the GFSes computed from the minimal global diagnoses coincides with the GFSes computed from the LFSes. ◇

In summary: The conditions in Proposition 5, 6, and 7 can be used to compute the GFS of a components based on the LFS computes in each agent.

## 4.4 *Ready Fault Status for Faulty, Suspected, and Normal* GFS

As with fault status, the readiness can also be partitioned into two types.

DEFINITION 5: *The fault status of component* c *is* globally ready *if it is ready with respect to set of minimal conflicts in the sets of present and possible future minimal conflicts in all agents.*

DEFINITION 6: *The fault status of component* c *is* locally ready *for agent* A *if it is ready with respect to the set of present and possible future minimal conflicts in agent* A.

Since components might be supervised by several agents, a component might be locally ready even though it is not globally ready, and vice versa. The global readiness of a component can be computed using the propositions given in this section. Similar to the computation of the local readiness, propositions are given for the cases where the GFS is faulty, suspected, and normal.

The strong relationship between a fault status that is faulty and its readiness shown in Proposition 2 also holds for global readiness.

PROPOSITION 8: *Let the* GFS *of component* c *be* faulty, *then it is globally ready.*

*Proof.* Follows from Proposition 2 and Proposition 5. □

The relationship between global readiness and the normal and suspected GFS is based on similar conditions as in Proposition 6 and 7.

PROPOSITION 9: *In agent* $A$*, let* $\Pi^A$ *be the present conflicts and let* $\Pi^f_A$ *be the possible future conflicts. For an agent* $A$ *where the* LFS *of* $c$ *is suspected, let*

$$\bar{\Pi}^A = \{\pi \in \Pi^A : c \in \pi \wedge (\forall \tilde{\pi} \in \bigcup_{\tilde{A} \in \mathcal{A} \setminus A} \Pi^{\tilde{A}} : \tilde{\pi} \not\subset \pi)\}. \tag{11}$$

*Let the* GFS *of component* $c$ *be* suspected*, then it is globally ready if and only if*

$$\left(\nexists A \in \mathcal{A} : (\exists \pi^f_A \in \Pi^f_A : \pi^f_A = \{c\})\right) \wedge \tag{12a}$$

$$\left(\exists \pi \in \bigcup_{A \in \mathcal{A}} \bar{\Pi}^A : (\forall \tilde{\pi} \in \bigcup_{A \in \mathcal{A}} \Pi^f_A : \tilde{\pi} \not\subset \pi)\right). \tag{12b}$$

*Proof.* The GFS is ready if and only if the GFS could not become faulty or normal for any future set of global diagnoses, Definition 5. The GFS can not become faulty if and only if (12a) holds. The GFS could not become normal if and only if (12b) holds. □

In an implementation, the proposition above could be simplified by limiting the set of agents that the existential quantifiers range over. In equation (12a), the not exist quantifier ranging over the set of all agents $\mathcal{A}$ can be limited to the set of agents where the LFS is not ready. In equation (12b), the exist quantifier ranging over the set of conflicts in all agents $\mathcal{A}$, can be limited to the set of agents where, the LFS in $A$ is suspected and ready, or where the LFS is normal and not ready.

PROPOSITION 10: *Let* $\Pi^f_A$ *be the possible future conflicts in agent* $A$*. Let the* GFS *of component* $c$ *be* normal*, then it is globally ready if and only if*

$$\forall A \in \mathcal{A} : (\forall \pi^f_A \in \Pi^f_A : c \in \pi^f_A \wedge (\exists \pi \in \bigcup_{\tilde{A} \in \mathcal{A}} \Pi^{\tilde{A}} : \pi \subset \pi^f_A)) \tag{13}$$

*Proof.* Follows from Proposition 7. □

Similar to Proposition 9, the set that the universal quantifier range over can be limited. In equation (13), the exist quantifier over the set of all agents $\mathcal{A}$ can be limited to the set of agents where, the LFS is suspected, or normal and not ready.

In summary: The conditions in Proposition 8, 9, and 10 can be used to decide if a components GFS is ready based on the LFS in the different agents.

# 5 COMPUTING THE FAULT STATUS AND ITS READINESS

The propositions in Section 3 can be used to design an algorithm that computes the fault status and the readiness for all components. A direct design of the algorithm loops, for each component, over all diagnostic tests and checks if any of the conditions in the propositions are fulfilled. Even though such a direct design will give the desired results, the algorithm require quite some processing power since the algorithm must loop over all tests for all components. This section will therefore design an efficient algorithm that computes the fault statuses and readiness without having to loop over all components and tests. The algorithm is denoted the FSR algorithm (fault status and readiness).

The fault status for all components and the readiness for the faulty components can efficiently be computed using a straightforward implementation of the propositions in Section 3. However, as can be seen from Proposition 3 and 4, an efficient implementation of the computation of the readiness for the suspected and the normal components require knowledge of which conflicts that are subsets of other conflicts. This ordering of the set of conflicts is a partially ordered set that here will be represented by a directed acyclic graph (DAG) (Harary, 1969). In the DAG, the relationship between the different conflicts will directly be available and this will improve the efficiency when computing the readiness for the suspected and normal components.

The FSR algorithm consists of the construction of the DAG, the update of the constructed DAG with the results from the diagnostic test, and the computation of the fault status and the readiness. The DAG is constructed off-line as described in the following section.

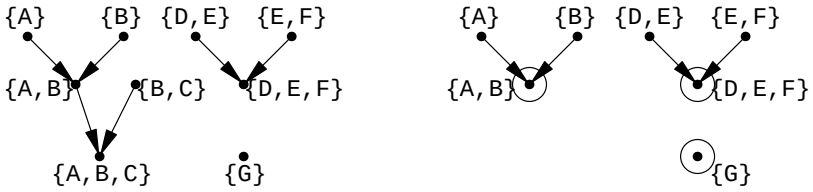
## 5.1 *The Construction of the Directed Acyclic Graph*

A DAG consists of vertices connected by directed edges. Here, each vertex will correspond to a conflict, while an edge will represent that a conflict is a subset of another conflict. The pair $(v_f, v_t)$ is used to denote that an edge is connected *from* vertex $v_f$ *to* vertex $v_t$.

The DAG is constructed such that a vertex $v$ is created for each unique conflict $\pi$ that could result from any diagnostic test. The conflict corresponding to vertex $v$ is denoted $\pi(v)$. Each vertex is further associated with a number $n(v)$ that is the number of tests with conflict $\pi$, and a mark $\mathrm{mark}(v)$ that is 0 if no test with conflict $\pi(v)$ has responded and 1 otherwise. Two vertices are connected by a directed edge $(v_f, v_t)$ if $\pi(v_f) \subset \pi(v_t)$ and a vertex $v$ does not exist such that

$\pi(v_f) \subset \pi(v) \subset \pi(v_t)$).

EXAMPLE 6: Consider a system with eight components and nine diagnostic tests. The diagnostic tests supervise the components and the corresponding conflicts are $\{A\}$, $\{B\}$, $\{A, B\}$, $\{B, C\}$, $\{A, B, C\}$, $\{D, E\}$, $\{E, F\}$, $\{D, E, F\}$, and $\{G\}$. Based on this information, the DAG in Figure 2(a) is created where each vertex has been labeled with its corresponding conflict. None of the vertices are marked since no test have been evaluated, and since each test has a unique conflict the number $n(v)$ is one for all vertices. ◇




(a) The orignial DAG. (b) The DAG updated with test results.

FIGURE 2: A DAG is constructed and updated.

## 5.2 *Updating the* DAG *Based on the Results from Tests*

After that a diagnostic test has been evaluated, the DAG should be updated with respect to the test results. If a test has responded, then a test whose conflict is a superset of the conflict for the responded test should be removed from the DAG, since it is now non-minimal. On the other hand, if a test did not respond then the corresponding vertex can be removed, unless the number $n(v)$ is greater than one. The update of the DAG is performed by Algorithm 1.

EXAMPLE 7: Continuation of Example 6. Assume that the diagnostic tests corresponding to the conflicts $\{A, B\}$, $\{D, E, F\}$, and $\{G\}$ have responded and the diagnostic test corresponding to the conflict $\{B, C\}$ has not responded. The DAG is updated using Algorithm 1. Due to the mark of the vertex for the conflict $\{A, B\}$, the vertex corresponding to conflict $\{A, B, C\}$ is removed. The resulting DAG is shown in Figure 2(b) where the vertices with a mark are circled. ◇

## 5.3 *Computing the Fault Statuses and Their Readiness*

After the DAG has been constructed and updated with the diagnostic test results, the fault statuses and the readiness can be computed

**Algorithm 1** Update DAG G with respect to the diagnostic test results.

**Input:** DAG $G = (V, E)$. Responded and not-responded diagnostic tests.

**Output:** An updated DAG $G = (V, E)$.

**if** the diagnostic test corresponding to vertex $v$ has responded **then**
  $V := V \setminus \{v_{rm} : v_{rm} \text{ reachable from } v\}$ [*Vertices to remove.*]
  $E := E \setminus \{(v_f, v_{rm}) : v_{rm} \text{ reachable from } v\}$ [*Edges to remove.*]
  $\text{mark}(v) := 1$ [*Set the mark.*]
**else**
  **if** $n(v) > 1$ **then**
    $n(v) := n(v) - 1$ [*Reduce the number.*]
  **else**
    $E := E \cup \{(v_f, v_t) : (v_f, v) \in E \wedge (v, v_t) \in E\}$ [*Add edges.*]
    $E := E \setminus \{(v_f, v) \in E\} \cup \{(v, v_t) \in E\}$ [*Remove edges.*]
    $V := V \setminus \{v\}$ [*Remove v.*]
  **end if**
**end if**

from the graph using Algorithm 2. In the algorithm, $X^C$ denotes the complement set of $X$ with respect to the set of components $\mathcal{C}$.

The algorithm first computes the sets of components whose fault status are faulty, suspected, and normal, which in the algorithm are the sets $F$, $S$, and $N$, respectively. For example, the components with fault status faulty are given by the vertices that have a mark and whose conflicts have a cardinality of one.

EXAMPLE 8: Continuation of Example 6. Consider first the components with fault status faulty. For the vertex $v$ corresponding to the conflict $\{G\}$, the $\text{mark}(v) = 1$, and $|\pi(v)| = 1$, resulting in $F := \{G\}$. The set of suspected components is computed to $S := \{A, B, D, E, F\}$, and the set of normal components is therefore $N := \{C\}$. ◇

The second part of the algorithm is the computation of the readiness for all components. The faulty and ready components are given directly by the set of faulty components, Proposition 6.

To compute if a suspected component is ready, it should be checked if the component could become faulty or normal in the future, using Proposition 3. First, if an unmarked vertex exists whose conflict has a cardinality of one and includes the component, then the fault status could become faulty. Second, if the component is included in all conflicts, corresponding to vertices that can reach one of the vertices resulting in the suspected component, then the component could not become normal. In both cases, it is sufficient to check the root vertices, since these includes the fewest number of components in its corre-

**Algorithm 2** Compute the fault status and readiness for all components.

**Input:** A DAG $G$ representing the diagnostic tests.

**Output:** A set $T$ including a tuple for each component in $\mathcal{C}$. Each tuple includes the fault status and readiness for a component.

1: $F := \{c \in \pi : v \in V \wedge \text{mark}(v) = 1 \wedge \pi(v) = \pi \wedge |\pi| = 1\}$ [*Faulty.*]
2: $S := \{c \in \pi : v \in V \wedge \text{mark}(v) = 1 \wedge \pi(v) = \pi \wedge |\pi| > 1\}$ [*Suspected.*]
3: $N := (F \cup S)^C$ [*Normal.*]
4: $R_S := \emptyset$, $x := S$ [*Ready and suspected* $R_S$*, possibly ready* $x$*.*]
5: **for each** $v \in V$ where $\text{mark}(v) = 1$ and $|\pi(v)| > 1$ **do**
6: **if** $v$ is a root **then**
7: $R_S := R_S \cup \pi(v)$, $x := x \setminus \pi(v)$ [*Ready.*]
8: **else**
9: **for all** $v_s \in V$ where $v_s$ is a root to $v$, $|\pi(v_s)| = 1$, and $\pi(v_s) \cap x \neq \emptyset$ **do** $x := x \setminus \pi(v_s)$ [*Not ready.*]
10: **for all** $c \in x \cap \pi(v)$ **do**, **if** $\forall v_s \in V$ where $v_s$ is a root to $v$ and $c \in \pi(v_s)$ **then** $R_S := R_S \cup \{c\}$, $x := x \setminus \{c\}$ [*Ready.*]
11: **end if**
12: **end for**
13: $R_N := N \setminus \{c \in \pi : v \in V \wedge \text{mark}(v) = 0 \wedge v \text{ is a leaf } \wedge \pi(v) = \pi\}$ [*Ready and normal* $R_N$*.*]
14: $R := F \cup R_S \cup R_N$ [*The set of ready components.*]
15: $T := \{\langle s, r\rangle_c : c \in \mathcal{C}, s = \text{faulty if } c \in F, s = \text{suspected if } c \in S, s = \text{normal if } c \in N, r = \text{ready if } c \in R, r = \text{not-ready if } c \in R^C\}$ [*Tuples.*]

sponding conflicts.

EXAMPLE 9: Continuation of Example 6. Initiate the set of ready suspected components to $R_S := \emptyset$, and the possibly ready components to $x := S$. Consider first the marked vertex $v$ corresponding to the conflict $\{A, B\}$, see Figure 2(b). The root vertices $\{A\}$ and $\{B\}$ show that neither $A$ nor $B$ is ready since they might in the future become faulty. They are therefore removed from the set of possibly ready components $x := \{D, E, F\}$. Consider now the marked vertex $v$ corresponding to the conflict $\{D, E, F\}$. The component $E$ is ready since it is included in all root vertices. The component is added to the ready components $R_S := \{E\}$ and removed from the possible ready components $x := \{D, F\}$. The result is that $E$ is ready while $D$ and $F$ are not. ◇

To check if a component with normal fault status is ready, it follows from Proposition 4 that it is only necessary to check the unmarked vertices that are leafs. The final step in the algorithm is to compute the tuples that also is the output from the algorithm.

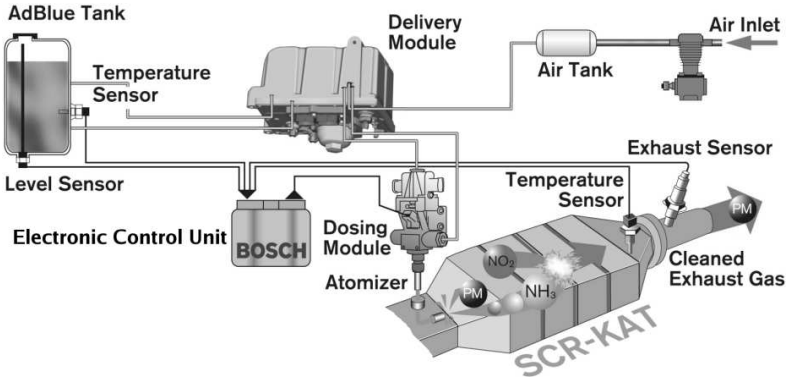



FIGURE 3: The SCR in the application (Bosch, 2006).

## 5.4 *The Correctness of the* FSR *Algorithm*

The correctness of the FSR algorithm is shown using Theorem 11.

THEOREM 11: *Let* $\Pi$ *be the set of present minimal conflicts and* $\Pi^f$ *the set of possible future conflicts. For each component, let the tuple be* $\langle s, r\rangle_c$ *where the fault status for component* $c$ *is* $s \in \{faulty, suspected, normal\}$ *and the readiness is* $r \in \{ready, not\ ready\}$*. Let the result from Algorithm 2 be* $T$*, then for each component* $c$*, a tuple* $\langle s_T, r_T\rangle_c \in T$ *exists where* $\langle s_T, r_T\rangle_c = \langle s, r\rangle_c$*.*

*Proof.* The correctness of F, S, and N follow from Definition 1. The correctness of the sets $R_S$ and $R_N$ are shown using Proposition 3 and 4 respectively. It follows from Proposition 2 and the correctness of $R_S$ and $R_N$ that R is the set of components whose fault statuses are ready. Output T is constructed correct parts and is therefore correct. □

## 5.5 *Extending the Algorithm to Distributed Systems*

The FSR algorithm computes the fault statuses and readiness for one agent. The algorithm can be extended using the propositions in Section 4 such that it computes the GFS and the global readiness in a distributed system. To compute the GFS and the global readiness, the part of each DAG that includes vertices pointing at vertices in another DAG would have to be transmitted to the other agents.

# 6 AUTOMOTIVE VEHICLE APPLICATION

The FSR algorithm has been evaluated on a part of the diagnostic system used in the heavy duty truck from Scania that is described in Section 4.1. The part consists of the selective catalytic reduction (SCR)

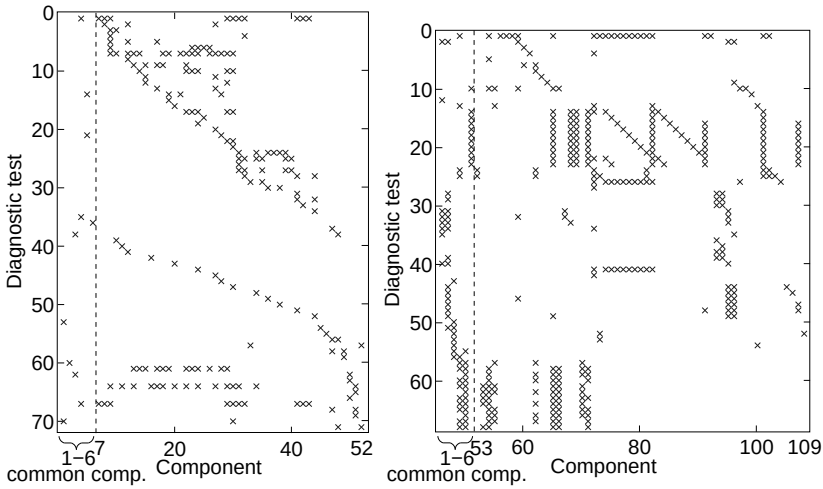



FIGURE 4: The isolation structure for the SCR (left) and the EMS (right).

system, which will be studied in detail, and the engine management system (EMS), which will be included to create a distributed system. The SCR lowers nitrous oxides from diesel engines and is used in new trucks from all European heavy duty vehicle manufacturers, see Figure 3 for a schematic overview. The main parts of the system are the SCR catalytic converter, the delivery module and the dosing module that deliver the reduction agent, and the ECU that controls and supervises the SCR. The EMS controls and supervises the engine. The control of the engine is closely related to the control of the SCR, and several components are therefore supervised both by the EMS and the SCR.

The diagnostic system in both the SCR and the EMS each consists of about 70 diagnostic tests that supervise about 50 and 60 components respectively. The isolation structures can be seen in Figure 4, where a cross $\times$ in row $i$ and column $j$ means that the diagnostic test $i$ supervises the component $j$ for abnormal behavior. The six first components in the isolation structures are supervised by both the diagnostic system in the EMS and in the SCR. If all diagnostic tests were single-component tests, then the isolation structures would, with a suitable ordering of the tests, be diagonal matrices. However, it can be seen that in this case, several plausibility tests exist and this makes the computation of the statuses more difficult.

The FSR algorithm first creates DAGs off-line based on the isolation structures, see Figure 5 for the SCR, and Figure 7 for the EMS. After the DAGs have been created, Algorithm 1 and 2 is used to compute the fault status and the readiness for all components.

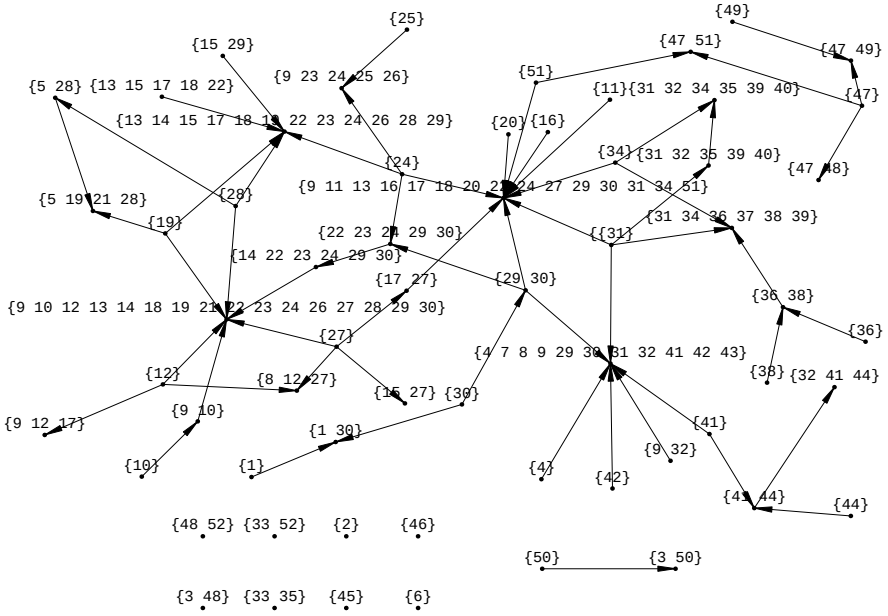



FIGURE 5: The original DAG for the SCR. The conflicts are displayed at the corresponding vertices.

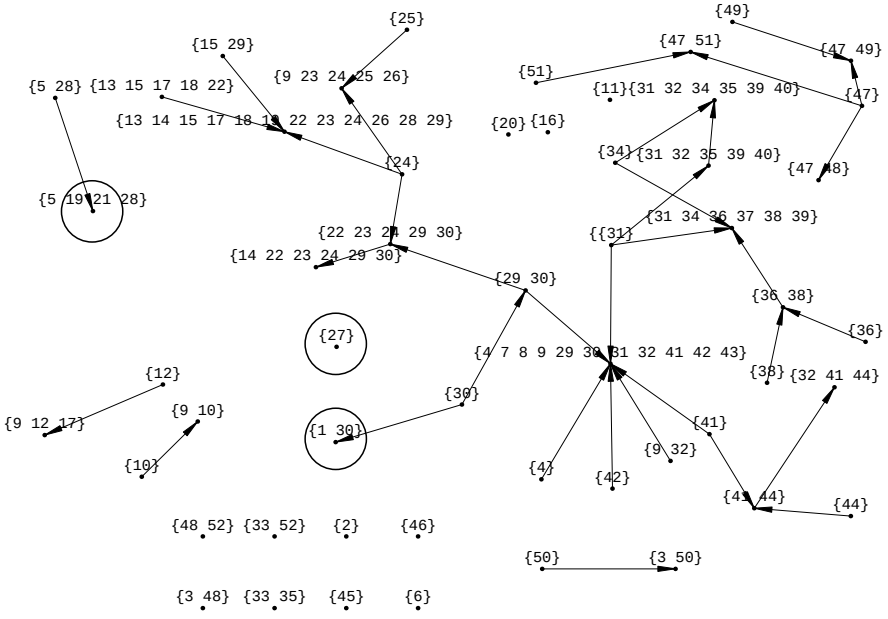



FIGURE 6: The updated DAG for the SCR with marked vertices circled.

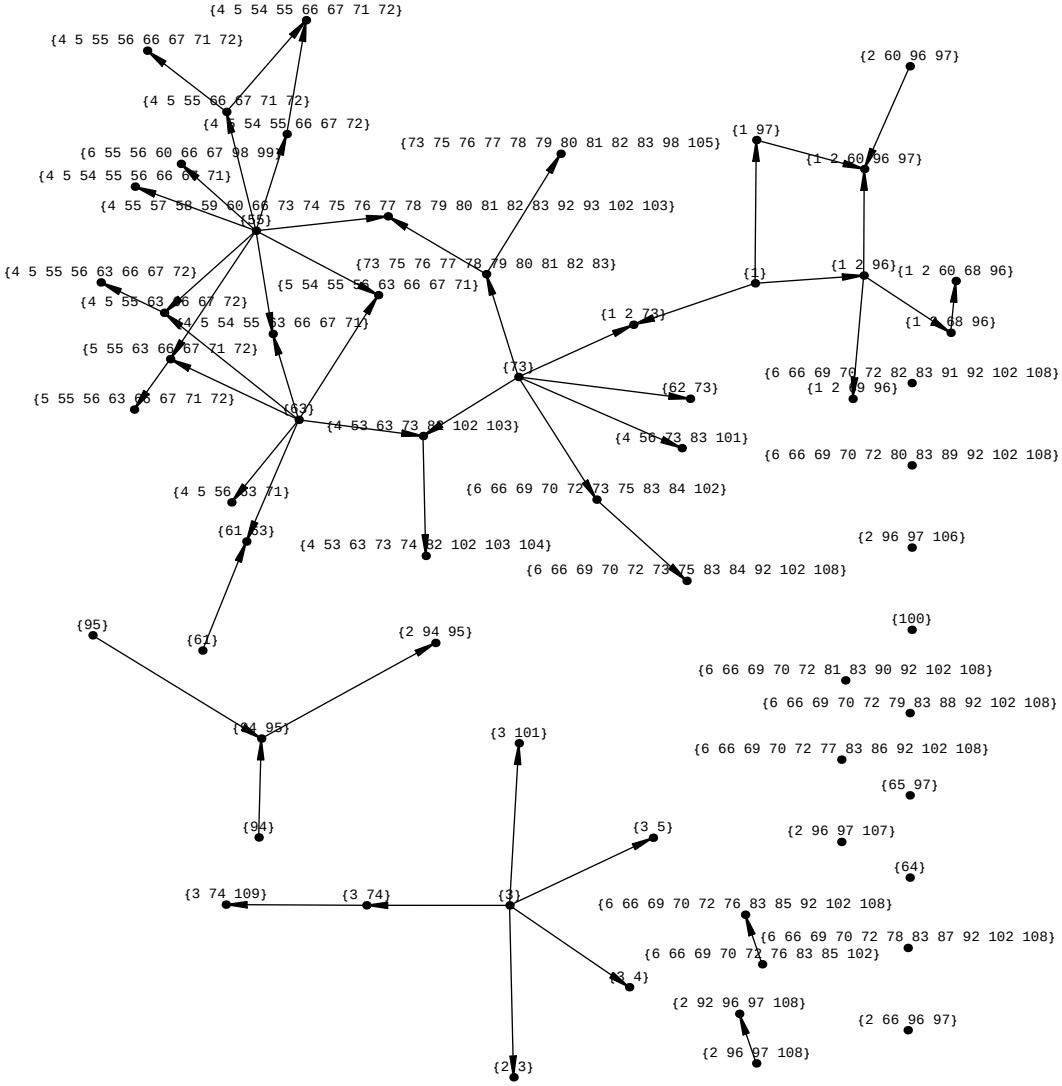



FIGURE 7: The original DAG for the EMS system. The conflicts are displayed at the corresponding vertices. Each vertex points at the vertices whose conflicts are supersets of its own conflict.

TABLE 1: Fault status and readiness for the SCR and the EMS.

| SCR | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | ... | 52 | 53–109 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Faulty | | | | | | | | | | | | | | | | | | | | | | | | | | | × | | | | | | | | | | | | |
| Suspected | × | | | | | × | | | | | | | | | | | | | | × | | × | | | | | | | × | | × | | | | | | | |
| Normal | | × | × | × | | × | × | × | × | × | × | × | × | × | × | × | × | × | | × | | × | × | × | × | × | | | | × | | × | × | × | × | × | × | × | × |
| Ready | | | | | × | | | | | | | | | | | | | | | | | | | | | | × | × | | | | | | | | | | | × |

| EMS | 1 | 2 | 3 | 4 | 5 | 6 | 7–52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | ... | 109 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Faulty | × | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Suspected | | × | × | × | | | | | × | × | × | | | | | × | | | | × | × | | | | × | × | | | | | | | | | | | |
| Normal | | × | | | | × | × | × | | | | × | × | × | × | × | × | × | × | | | | | | | | × | × | × | | × | × | × | × | × | × | × | × |
| Ready | × | | | | | | × | | | | | | | | | | | | | | | | | | × | | | | | | | | | | | | |

## 6.1 *Computation of Fault Status and Readiness*

To test the FSR algorithm, the three components numbered 1, 5, and 27 are affected such that they are behaving abnormally. Component 1 and 5 are supervised by both agents while 27 is only supervised by the SCR. The diagnostic tests corresponding to the conflicts $\{27\}$, $\{1, 30\}$, and $\{5, 19, 21, 28\}$ have been evaluated and responded, and the tests corresponding to conflicts $\{1\}$, $\{19\}$, and $\{28\}$ have been evaluated but not responded. Notice that the test supervising component 1 has made a missed detection, while the other two tests have made a correct detection. In the EMS, all tests that supervise any of the abnormal components have been evaluated and responded. No other tests have been evaluated. The DAGs are updated using Algorithm 1 resulting in the DAG shown in Figure 6 for the SCR, where the vertices corresponding to the responded diagnostic tests are circled. Notice that several vertices have been removed from the DAG. The corresponding DAG for the EMS is shown in Figure 8.

The fault status and the readiness are computed for all components using Algorithm 2 and the result can be seen in Table 1. Especially, in the SCR, component 27 is faulty and ready, components 1, 19 and 21 are suspected and not ready, and components 5 and 28 are suspected and ready.

The minimal diagnoses in the SCR have been computed for reference, resulting in $\{1, 5, 27\}$, $\{1, 19, 27\}$, $\{1, 21, 27\}$, $\{1, 27, 28\}$, $\{5, 27, 30\}$, $\{19, 27, 30\}$, $\{21, 27, 30\}$, and $\{27, 28, 30\}$. Using Proposition 1, it can be seen that the minimal diagnoses coincide with the fault statuses.

## 6.2 *Comparison Against a Direct Algorithm*

The design of the FSR algorithm is motivated by efficiency compared to a direct implementation of the propositions in Section 3. To verify this motivation, the FSR algorithm will here be compared to such

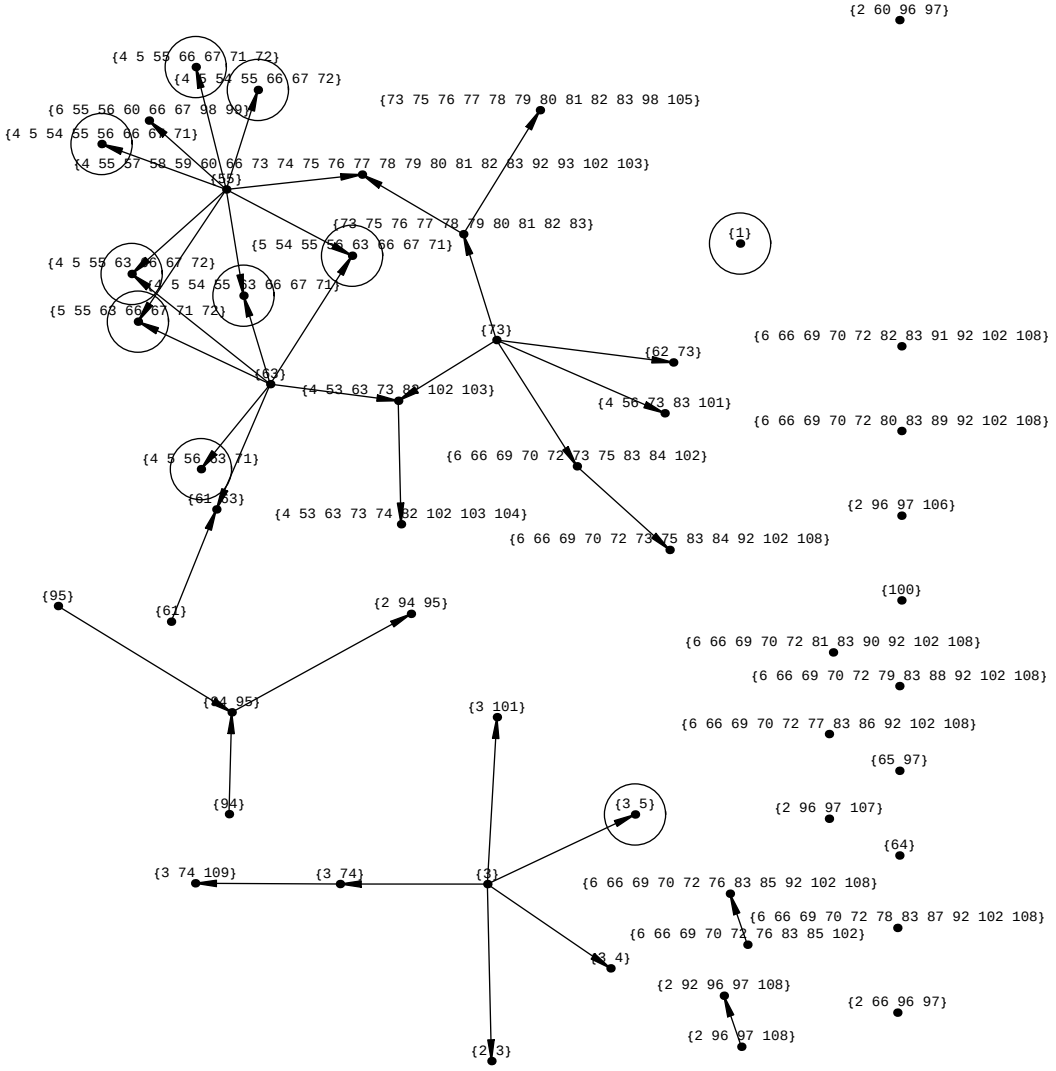



FIGURE 8: The updated DAG for the EMS. Vertices that are marked are circled.

a direct implementation. In the direct implementation, the universal quantifiers in the propositions are implemented as for-loops and is given as Algorithm 3.

**Algorithm 3** A direct implementation for the computation of fault status and readiness.

**Input:** A set of responded minimal conflicts $\Pi$.

**Output:** The sets $F$, $S$, $N$, and $R$ that are the faulty, suspected, normal, and ready components, respectively.

1: $F := \{c : \pi = \{c\} \wedge \pi \in \Pi\}$ [*Faulty.*]
2: $S := \{c \in \pi : |\pi| > 1 \wedge \pi \in \Pi\}$ [*Suspected.*]
3: $N := (F \cup S)^C$ [*Normal.*]
4: $\bar{S}_1 := \{c : \pi^f \in \Pi^f \wedge \pi^f = \{c\}\}$ [*In S and not ready.*]
5: $\bar{S}_2 := \{c \in \bar{\pi} \in \bar{\Pi} \subseteq \Pi^f : c \notin \bar{\pi} \wedge (\forall \pi \in \{\pi \in \Pi : c \in \pi\} : \bar{\pi} \subset \pi)\}$ [*In S and not ready.*]
6: $\bar{N} := \{c \in \pi^f \in \Pi^f : (\forall \pi \in \Pi : \pi \not\subset \pi^f)\}$ [*In N and not ready.*]
7: $R := F \cup (S \backslash \bar{S}_1 \backslash \bar{S}_2) \cup (N \backslash \bar{N})$ [*Ready.*]

The set of minimal conflicts is assumed to be known in the direct implementation, i.e. the non-minimal conflicts have been removed. The two methods will be compared by counting the number of non-trivial operations needed to compute the result. In Algorithm 1, each removal or addition of one edge or one vertex counts as one operation. In Algorithm 2 and in the direct implementation, each cardinality check of a conflict, union, intersection, and set removal counts as one operation.

To compute the fault statuses and the readiness for the SCR when affected by abnormal components 1, 5, and 27, the number of needed operations is 574 for the direct algorithm but only 118 for the FSR algorithm, a reduction in processor load with 80 %. To further compare the algorithms, the number of components in the abnormal mode has been varied and the mean number of needed operations has been computed, see Figure 9 for the SCR and Figure 10 for the EMS. For one, two, and three abnormal components, the results are exact, while for the higher number of abnormal components, the mean of one thousand different sets of abnormal components are used for each size. It can be seen in the figure that the mean number of operations needed with the FSR algorithm is lower than when the direct implementation is used, except for the unreasonable sizes where more than 50 components are abnormal. In fact, for all single, double, and triple abnormal components, the FSR algorithm is more efficient than the direct algorithm. In mean, the reduction in processor load is 80 % to 90 % for one to five abnormal components. For the EMS, the reduction in processor load is around 90 % for one to five abnormal components.

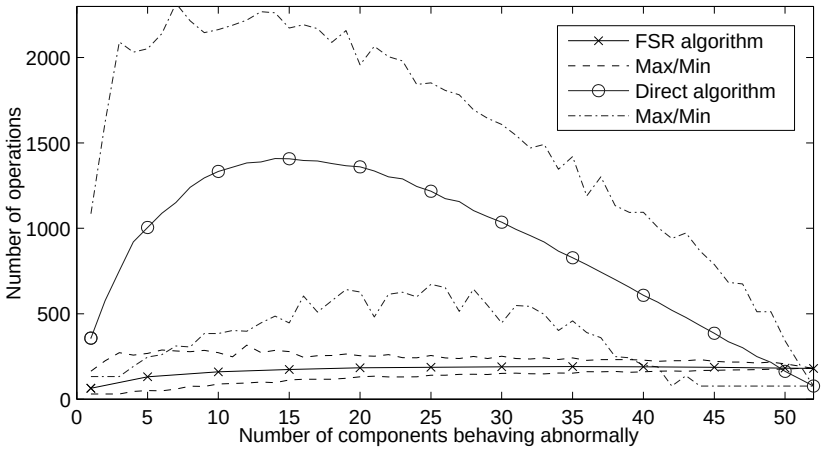



FIGURE 9: The mean number of operations needed to compute the fault statuses and their readiness in the SCR.

The complexity of the FSR algorithm is so low that it is always possible to compute the results with a reasonable number of operations. This is in contrast to the computation of the minimal diagnoses, where this is not always possible. If for example components numbered 1 and 60 are behaving abnormally, then about 60 000 operations is needed to compute the resulting 300 minimal diagnoses in the EMS. If also component 70 is behaving abnormally, then the problem is intractable on-line since over 20 000 000 non-trivial operations are needed resulting in over 7 000 minimal diagnoses after over two hours of computations on a desktop PC. However, the FSR algorithm requires about 100 operations in both cases and is therefore tractable.

## *6.3 Global Fault Status and Readiness*

Since the local fault status (LFS) and the local readiness has been computed for each component for both agents, these can be used to state the global fault status (GFS) and the global readiness. Component numbered 1 is for example locally suspected in the SCR while being locally faulty and ready in the EMS.

Using Proposition 5 and 8, it can be computed that the GFS of component 1 is faulty and ready, since the LFS is faulty in the EMS. The control system in the SCR can now use the knowledge that component 1 certainly is behaving abnormally. Proposition 6 shows that the GFS of component 5 is suspected since an agent exists where the LFS is suspected, and no conflict exists that is a subset of the conflict that made the LFS suspected, i.e. the conflict $\{5, 19, 21, 28\}$, and does not

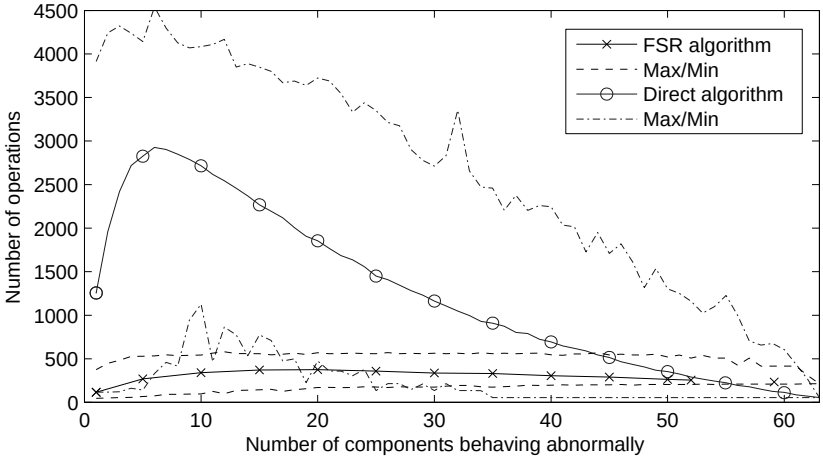



FIGURE 10: The mean number of operations needed to compute the fault statuses and their readiness for different number of abnormal components in the EMS.

include 5 or have a cardinality of one. Further, Proposition 9 shows that 5 is ready since only the vertex corresponding to conflict $\{5, 28\}$ points at the marked vertex including component 5. Notice that the information that has to be transfered from the EMS to the SCR is that any vertex in the EMS does not exist that points at the vertex with the conflict $\{5, 19, 21, 28\}$. The control system in the EMS could for example use the knowledge that component 5 is globally ready to choose an appropriate fault tolerant control strategy.

# 7 DIAGNOSTIC TESTS THAT RESULTS IN READY STATUS

One of the objectives for a diagnostic system is to achieve readiness for the fault status of a component, and the problem of which diagnostic tests to evaluate to achieve readiness will here be studied.

## *7.1 Meaningful Diagnostic Tests*

If the fault status of a component is not ready, then, by definition, diagnostic tests exist that could change the fault status, these diagnostic tests are denoted the meaningful diagnostic tests.

DEFINITION 7 (Meaningful diagnostic tests): *Let* $\Pi$ *be the set of conflicts corresponding to a set of diagnostic tests and partition this set into* $\Pi_{add}$ *and*

$\Pi_{remove}$. *The set of diagnostic tests is* meaningful *for component* c *if the addition of* $\Pi_{add}$ *to the set of present conflicts and the removal of* $\Pi_{removal}$ *from the set of future conflicts would result in a change in the fault status or the readiness of component* c.

Similar to diagnoses and conflicts, a set of meaningful diagnostic tests is minimal if there is no proper subset that is a set of meaningful diagnostic tests.

From the definition follows that the status of a component is ready if and only if no set of meaningful diagnostic tests exists. Different sets of diagnostic tests are meaningful when the status of a component is faulty, normal, or suspected, as shown by the following three propositions.

PROPOSITION 12: *Let the status of component* c *be* faulty, *then no sets of meaningful diagnostic tests exist for component* c.

*Proof.* Follows directly from Proposition 2. □

PROPOSITION 13: *Let the status of component* c *be* suspected. *The minimal sets of meaningful diagnostic tests for component* c *are the sets of diagnostic tests that correspond to the minimal sets of conflicts in the set*

$$\left\{\bar{\Pi} \subseteq \Pi^{f} : \left(\forall \pi \in \{\pi \in \Pi : c \in \pi\} : \left(\exists \bar{\pi} \in \bar{\Pi} : (c \notin \bar{\pi} \wedge \bar{\pi} \subset \pi)\right)\right)\right\} \tag{14a}$$

*and to the sets of conflicts*

$$\left\{\{\pi^{f}\} : \pi^{f} \in \Pi^{f} \wedge c \in \pi^{f} \wedge (\nexists \pi_2^{f} \in \Pi^{f} : \pi_2^{f} \subset \pi^{f} \wedge c \notin \pi_2^{f})\right\}. \tag{14b}$$

*Proof.* Proposition 3 gives both some diagnostic tests that are meaningful in themselves, and sets of conflicts that only are meaningful if all diagnostic tests in the set are evaluated. Equation (7b) corresponds to (14a). Equation (14b) corresponds to the case where component c becomes ready and in some cases also faulty (7a). □

A set of conflicts in (14a) changes the suspected status to normal, while a conflict in a set in (14b) changes the suspected status to faulty or the fault status to ready. The next proposition states which sets that are meaningful when the fault status of a component is normal.

PROPOSITION 14: *Let the status of component* c *be* normal, *then the minimal sets of meaningful diagnostic tests are the sets of diagnostic tests that correspond to the sets of conflicts*

$$\left\{\{\pi^{f}\} : \pi^{f} \in \Pi^{f} \wedge c \in \pi^{f} \wedge (\nexists \pi \in \Pi : \pi \subset \pi^{f})\right\} \tag{15a}$$

*and to the minimal sets of conflicts in the set*

$$\left\{\bar{\Pi} \subseteq \Pi^{f} : (\forall \pi^{f} \in \Pi^{f} \backslash \bar{\Pi} : c \notin \pi^{f})\right\}. \tag{15b}$$

**Algorithm 4** Compute meaningful diagnostic tests.

**Input:** The DAG $G$. Suspected $S$, normal $N$, and ready $R$.

**Output:** Sets of sets of meaningful diagnostic tests including the minimal sets of meaningful diagnostic tests. $M_{s2n}$ meaningful for changing suspected components to normal. $M_{s2rf}$ : suspected to ready or faulty. $M_{n2sf}$ : normal to suspected or faulty. $M_{n2r}$: normal to ready.

1: **for each** $c \in S \cap R^C$ **do**
2: $X := \emptyset$.
3: **for each** $v \in V$ where $c \in \pi(v) \wedge mark(v) = 1$ **do**
4: $x := \{\bar{v} \in V : (\bar{v}$ can reach $v) \wedge c \notin \pi(\bar{v})\}$.
5: **if** $x = \emptyset$ **then** break for-loop **else** $X := X \cup x$.
6: **end for**
7: $M_{s2n}(c) :=$ cross-product$(X)$.
8: $M_{s2rf}(c) := \{\bar{v} \in V : (\bar{v}$ is a source$) \wedge c \in \pi(\bar{v}) \wedge mark(\bar{v}) = 0\}$.
9: **end for**
10: **for each** $c \in N \cap R^C$ **do**
11: $X := \emptyset$.
12: **for each** $v \in V$ where $c \in \pi(v) \wedge mark(\pi) = 1$ **do**
13: $x := \{\bar{v} \in V : (v$ reachable from $\bar{v}) \wedge c \notin \pi(\bar{v})\}$.
14: **if** $x = \emptyset$ **then** break for-loop **else** $X := X \cup x$.
15: **end for**
16: $M_{n2r}(c) :=$ cross-product$(X)$.
17: $M_{n2sf}(c) := \{\{v\} : v \in V \wedge c \in \pi(v) \wedge mark(\bar{v}) = 0\}$.
18: **end for**

*Proof.* Proposition 4 gives (15a). Equation (15b) is the case where $c$ becomes ready. □

In summary: Depending on the fault status of a component, the conditions in Proposition 12 to 14 can be used to calculate the sets of meaningful diagnostic tests. However, a direct implementation of the propositions might not be efficient, and the DAG representation that is used in Section 5 when computing the fault status and its readiness will also be used to compute the sets of meaningful diagnostic tests.

## 7.2 *Computing the Meaningful Diagnostic Tests*

Using the same DAG as used by Algorithm 2 in this paper, Algorithm 4 computes the sets of meaningful diagnostic tests for different components. The sets that are computed are $M_{s2n}(c)$ and $M_{s2rt}(c)$ for components that are suspected and not-ready, and the sets $M_{n2r}(c)$ and $M_{n2sf}(c)$ that are computed for components that are computed for components that are normal and not-ready. The set $M_{s2n}(c)$ is

TABLE 2: Fault status and readiness for the SCR after the meaningful tests have been evaluated.

| SCR | 1 | | | | | 6 | 7 | | | 10 | | | | 15 | | | | 20 | | | | | 25 | 27 | | 30 | | | | | 35 | ... | 52 | 53–109 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Faulty | | | | | | | | | | | | | | | | | | | | | | | | × | | | | | | | | | |
| Suspected | × | | | | | × | | | | | | | | | | | | | | | | | | | × | × | | | | | | | |
| Normal | | × | × | × | | | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | | | | × | | × | × | × | × | × | × | × |
| Ready | × | | | | | × | | | | | | | | | | | | × | | | | | | × | × | × | | | | | | | × |

meaningful for changing the suspected and not-ready component c to normal and $M_{s2rf}(c)$ for changing to ready or faulty. The set $M_{n2sf}(c)$ is meaningful for changing normal and not-ready to suspected or faulty and $M_{n2r}(c)$ for changing to ready.

### 7.3 *Automotive Application*

This section is a continuation of Section 6.1 where components numbered 1, 5, and 27 were abnormal.

In Section 6.1 it is found that components 1, 19, 21, and 30 were suspected but not ready. To change the fault status or the readiness of these components, the sets of meaningful diagnostic tests are computed using Algorithm 4. Minimal meaningful diagnostic tests for the suspected components are the tests with conflicts $\{30\}$ and $\{5, 28\}$. Assume that these two diagnostic tests are evaluated and the test with conflict $\{5, 28\}$ has responded while the test with conflict $\{30\}$ has not responded. An update of the DAG in Figure 6 on page 113 with this new information results in the DAG shown in Figure 11. The fault status and the readiness is then computed for every component using Algorithm 2 and the result is shown in Table 2. Components 5, 21, and 30 are now ready and this shows that it was meaningful to evaluate the specific diagnostic tests.

A similar analysis for the EMS shows that eleven diagnostic tests with different conflicts are meaningful for the suspected components. The update of the DAG shown in Figure 8 on page 113 results in the DAG shown in Figure 12.

## 8 SCHEDULING MEANINGFUL TESTS

Given that some diagnostic tests are meaningful with respect to some specific set of components, the question of in which order that these meaningful diagnostic tests should be scheduled for evaluation arises. One approach is to first schedule the diagnostic test whose evaluation makes the most components to become ready, and then the other tests

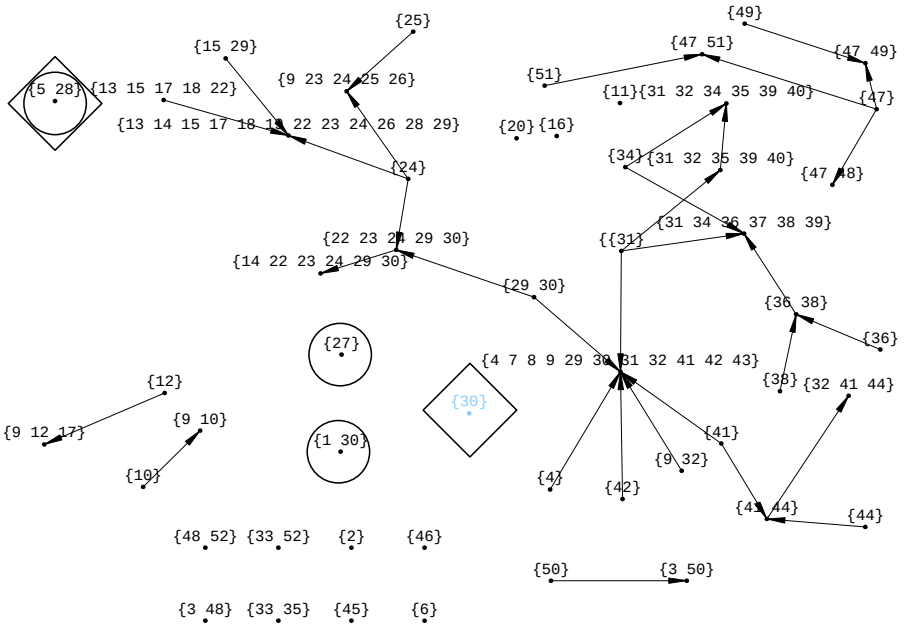



FIGURE 11: The updated DAG G for the SCR with conflicts displayed at the corresponding vertex. Diagnostic tests that have been evaluated after they were considered meaningful for the suspected components are marked with a diamond.

in descending order. When computing the schedule, it should be considered that a diagnostic test could both respond and not respond, since some components could become ready in both cases. This section will design a strategy that computes a best schedule based on the idea introduced above. By evaluating the meaningful tests according to the best schedule, the number of tests that has to be evaluated can be reduced to a minimum, and thereby for example reducing the usage of computational resources.

In the general case, sets of meaningful diagnostic tests exist where two or more tests must be evaluated for them to be meaningful. Here only sets of single meaningful tests will be considered. However, the designed strategy can be extended to manage larger sets.

## *8.1 Strategy for Scheduling the Meaningful Tests*

The section aims at defining a value of the gain for each set of meaningful diagnostic tests. The value represents the number of components that will become ready when the meaningful diagnostic test is

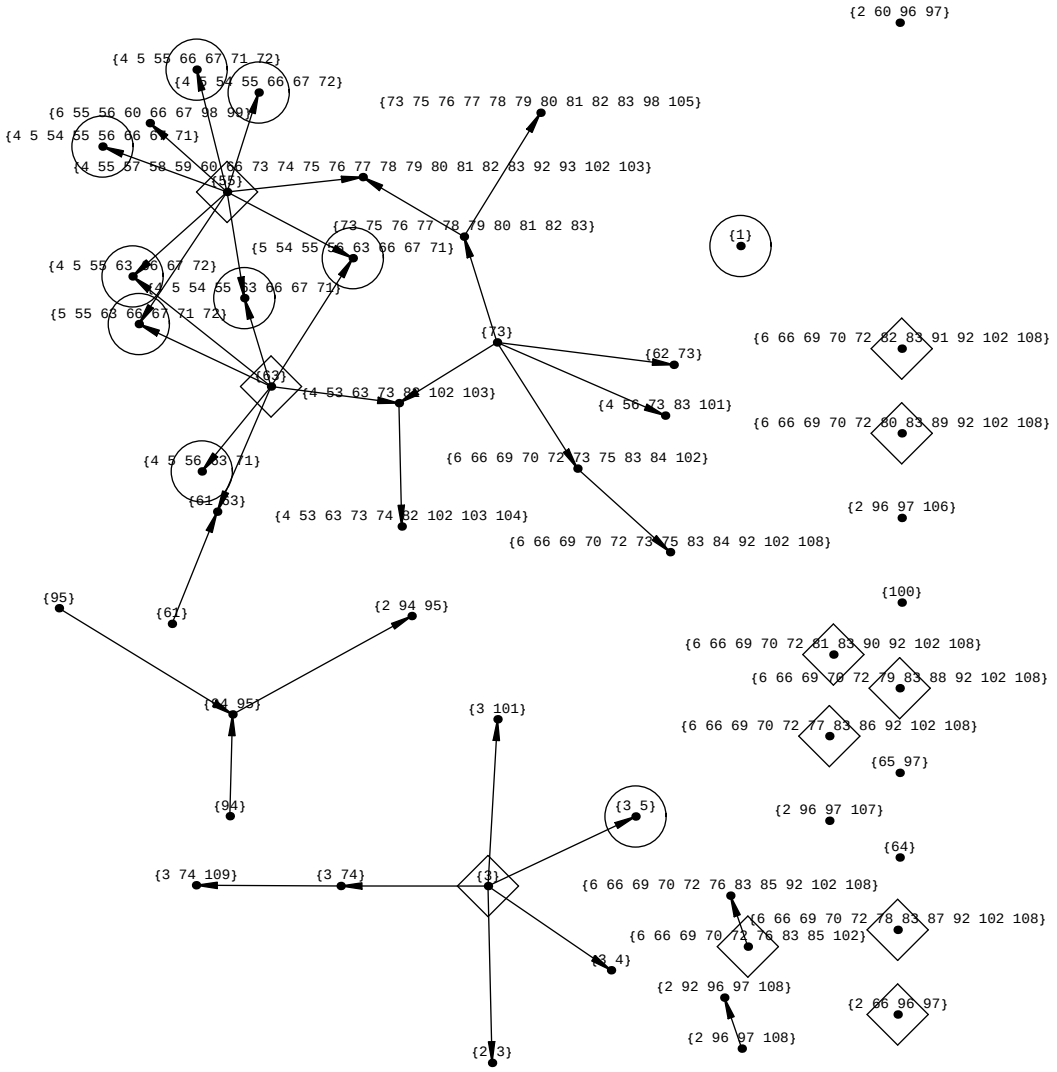



FIGURE 12: The updated DAG G for the EMS with conflicts displayed at the corresponding vertex. Vertices that have a mark are circled. Meaningful diagnostic tests are marked with a diamond.

evaluated. The sets of meaningful diagnostic tests will then be scheduled such that the test with highest gain is evaluated first, and the rest follows in descending order of gain.

First, let the gain when evaluating a meaningful test $m$ be

$$\sigma_R(m) := |R(m \text{ did respond})| - |R(\text{before } m \text{ was evaluated})|$$
$$\sigma_{\neg R}(m) := |R(m \text{ did not respond})| - |R(\text{before } m \text{ was evaluated})|$$

for the cases where $m$ respond and where it does not respond respectively. The set $R$ is the set of components whose status is ready. The gains in readiness $\sigma_R(m)$ and $\sigma_{\neg R}(m)$ can be computed using Algorithm 2. Above, the total number of components that become ready is considered. An alternative to this would be to only consider how many components in some specific set of components that becomes ready, for example how many of the suspected components become ready when a test is evaluated.

Let $T_R$ and $T_{\neg R}$ be the set of tests that have been evaluated and responded and evaluated and not responded, where the corresponding conflicts are minimal. Assume that the probability $P(m|T_R \wedge T_{\neg R})$ that a diagnostic test responds given a set of responded tests and not responded tests is known. A reasonable value of the gain for the meaningful diagnostic test $m$ is then

$$\xi(m) = P(m|T_R \wedge T_{\neg R})\sigma_R(m) + (1 - P(m|T_R \wedge T_{\neg R}))\sigma_{\neg R}(m).$$

The gain is the expected value of the gain when evaluating test $m$. The meaningful test with highest $\xi(m)$ should be evaluated first since this would most probably give the highest gain in number of ready components.

**Probability for a test to respond**

What is the value of $P(m|T_R \wedge T_{\neg R})$? Generally, a good approximation of this number is difficult to find due to difficulties in deciding a-priori probabilities for the abnormal behavior of a component, diagnostic test detection, diagnostic test false alarm, etc. If these probabilities are available, then the probability $P(m|T_R \wedge T_{\neg R})$ can directly be computed using for example a Bayesian network (Jensen and Nielsen, 2001). Unfortunately, the Bayesian computations require that at least the set of minimal cardinality diagnoses have been computed, see Appendix A. The FSR algorithm is motivated by the fact that it requires low processing power and the requirement can therefore not be fulfilled, since the computation of the minimal cardinality diagnoses might require a substantial amount of processing power, see for example Section 6.2.

Due to the difficulties described above, a heuristic value of the probability $P(m|T_R \wedge T_{\neg R})$ will here be derived. The heuristic value

will be given for two different cases. The first case is when the diagnostic test $m$ supervises some component that is also supervised by some of the tests in $T_R$. The second is when this is not the case. It will be assumed that the tests that supervise a component that some other test have found to be suspected is more likely to respond than the tests that does not supervise such a component.

For a test $m$ in the first case, the variable

$$\lambda_{\text{supervise}}(m) := \frac{|\pi(m) \cap \bigcup_{t \in T_R} \pi(t)|}{|\bigcup_{t \in T_R} \pi(t)|}$$

will be used. The value is reasonable since it increases with the number of components supervised by both the test $m$ and the responded tests in $T_R$. The set $\bigcup_{t \in T_R} \pi(t)$ includes all components supervised by any responded test. If a test supervises more components in the set $\bigcup_{t \in T_R} \pi(t)$ than some other test, then it is more probably that the first test responds. The denominator is a normalization and can be left out.

For a test $m$ in the second case, the variable

$$\lambda_{\neg\text{supervise}}(m) := \frac{|\pi(m)|}{|\mathcal{C}|}$$

will be used instead. The value is reasonable since it increases if the number of supervised components increases.

### The gain in evaluating a test

The value of the gain can now be stated given the two heuristic values. For a test that supervise some component also supervised by a test in $T_R$, the value of the gain is

$$\xi_{\text{supervise}}(m) := \lambda_{\text{supervise}}(m)\sigma_R(m) + (1 - \lambda_{\text{supervise}}(m))\sigma_{\neg R}(m).$$

For a test that does not supervise such a component, the gain is

$$\xi_{\neg\text{supervise}}(m) := \lambda_{\neg\text{supervise}}(m)\sigma_R(m) + (1 - \lambda_{\neg\text{supervise}}(m))\sigma_{\neg R}(m).$$

### Test Evaluation Strategy

The following strategy is proposed: Compute the sets of minimal meaningful tests. Eventually limit the search to the set of tests that are meaningful for some chosen subset of not-ready components, for example the suspected components. Compute $\xi_{\text{supervise}}(m)$ for each meaningful test $m$ that supervise a component supervised by a test in $T_R$. The meaningful diagnostic test with highest $\xi_{\text{supervise}}(m)$ is scheduled first for evaluation and the rest of the meaningful tests then follows in descending order. For the rest of the meaningful diagnostic tests,

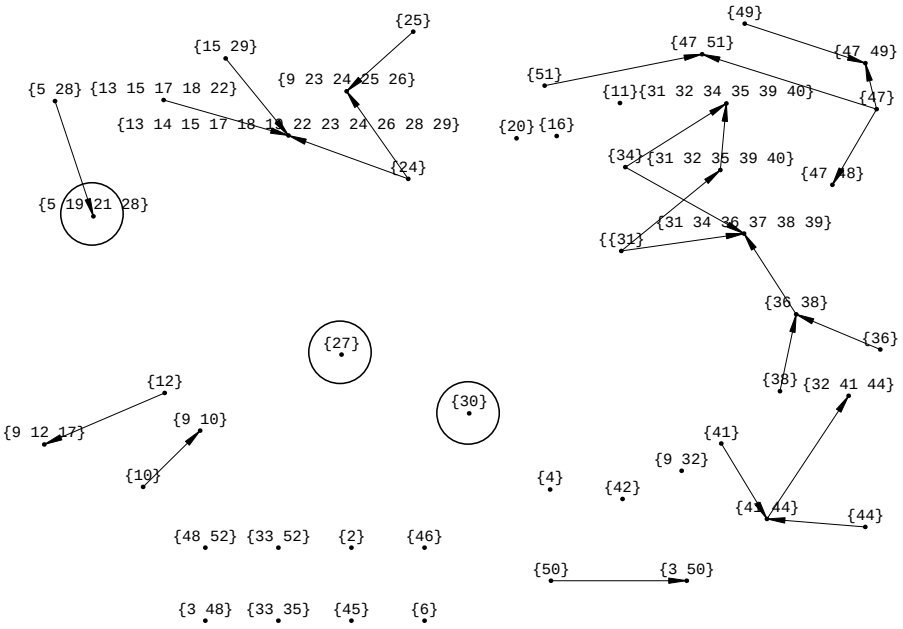



FIGURE 13: The updated DAG for the SCR when the diagnostic test with conflict $\{30\}$ have reacted.

compute $\xi_{\text{supervise}}(m)$ and the schedule these tests in descending order and schedule these tests after the other tests. After the first test in the schedule has been evaluated, a new search can be performed for a new schedule of the meaningful diagnostic test. By evaluating the meaningful tests in the best order, the number of tests that has to be evaluated can be reduced to a minimum, and thereby for example reducing the usage of computational resources.

## *8.2 Automotive Application*

For the application in Section 6, the diagnostic tests with conflicts $\{30\}$ and $\{5, 28\}$ were computed to be meaningful for the suspected components in the SCR, see Section 7. How should these two tests be scheduled to fastest give the most number of ready components?

For the diagnostic test with conflict $\{30\}$, the gain is $\sigma_R(\{30\}) = 8 - 3 = 5$ if the test respond and $\sigma_{\neg R}(\{30\}) = 2$ otherwise. The large number of ready components for the responded case arise since several components with normal fault status becomes ready, see Figure 13 where several vertices have been removed in the DAG since the test has responded. For the diagnostic test with conflict $\{5, 28\}$, the gains are

$\sigma_R(\{5,28\}) = 1$ and $\sigma_{\neg R}(\{5,28\}) = 2$. The weight for the two tests are $\lambda_{\text{supervise}}(\{30\}) = \frac{1}{6}$ and $\lambda_{\text{supervise}}(\{5,28\}) = \frac{2}{6}$ respectively. The values of the gain are therefore

$$\xi_{\text{supervise}}(\{30\}) = \frac{1}{6}5 + \frac{5}{6}2 = 2.5$$
$$\xi_{\text{supervise}}(\{5,28\}) = \frac{2}{6}1 + \frac{4}{6}2 \approx 1.7.$$

To conclude, when deciding in which order to evaluate the two meaningful diagnostic tests corresponding to the conflicts $\{30\}$ and $\{5,28\}$, it is best to first schedule the test with conflict $\{30\}$ and then the test with conflict $\{5,28\}$, since this strategy would in mean lead to a 2.5 ready components while the opposite strategy would in mean only lead to 1.7 ready components.

Above, the total number of normal components that have become ready after the evaluation of a test has been considered. If only the number of suspected components that become ready is considered, then the corresponding values are $\sigma_R(\{30\}) = 4-2 = 2$, $\sigma_{\neg R}(\{30\}) = 2$, $\sigma_R(\{5,28\}) = 2$, and $\sigma_{\neg R}(\{5,28\}) = 2$. The values are reduced compared to the case studied above. The values of the gain are in this case

$$\xi_{\text{supervise}}(\{30\}) = \frac{1}{6}1 + \frac{5}{6}1 = 1.0$$
$$\xi_{\text{supervise}}(\{5,28\}) = \frac{2}{6}1 + \frac{4}{6}2 \approx 1.7.$$

In contrast to the case above, it is best to first scheduled the test with conflict $\{5,28\}$ and then the other test.

TABLE 3: Weights for the meaningful diagnostic tests in the EMS.

| Conflict for the corresponding test | $\sigma_R$ | $\sigma_{\neg R}$ | $\lambda_{\texttt{high}}$ | $\xi_{\text{supervise}}$ |
|---|---|---|---|---|
| $\{55\}$ | 8 | 9 | 1/14 | 8.9 |
| $\{63\}$ | 3 | 5 | 1/14 | 4.9 |
| The other meaningful tests | 11 | 1 | 2/14 | 2.4 |
| $\{3\}$ | 2 | 2 | 1/14 | 2.0 |
| $\{6,66,69,70,72,76,83,85,102\}$ | 9 | 0 | 2/14 | 1.3 |
| $\{2,66,96,97\}$ | 4 | 0 | 1/14 | 0.3 |

For the EMS system, eleven different tests are meaningful for the suspected components, see Figure 12 where the meaningful tests are marked with diamonds. The gains and weights for these tests are shown in Table 3. As can be seen in the table, the diagnostic test corresponding to conflict $\{55\}$ should first be scheduled since this will in mean give 8.9 ready components. The next best strategy is to evaluate the test corresponding to conflict $\{63\}$ that in mean would lead

to 4.9 ready components. As with the SCR, the weights can be computed considering only the suspected components that become ready. The corresponding values are shown in Table 4. In contrast to the SCR, the same test, as when all ready components were considered, has the highest value of the gain. In this case, 7.6 components will in mean become ready.

TABLE 4: Weights for the meaningful diagnostic tests in the EMS when only the suspected components that become ready are considered.

| Conflict for the corresponding test | $\sigma_R$ | $\sigma_{\neg R}$ | $\lambda$ | $\xi_{\text{supervise}}$ |
|---|---|---|---|---|
| $\{55\}$ | 3 | 8 | 1/14 | 7.6 |
| $\{63\}$ | 1 | 5 | 1/14 | 4.7 |
| $\{3\}$ | 1 | 2 | 1/14 | 1.9 |
| The other meaningful tests | 1 | 0 | 2/14 | 0.14 |
| $\{2, 66, 96, 97\}$ | 1 | 0 | 1/14 | 0.07 |

### *8.3 Other Strategies*

The strategy designed above shows how the meaningful tests shall be scheduled if the objective is to gain the highest number of ready components. However, other objectives might be considered. If for example the diagnostic system is interested in a component c for which a diagnostic test with a conflict $\pi = \{c\}$ exists, then this test should probably be evaluated first, since this leads to fault status faulty and to readiness if the test responds.

Further, if the status for component c is suspected, then the ordering depends on if it is most important to find that the status is faulty or if it is most important to return the status to normal. If faulty is prioritized, then evaluate those diagnostic tests that fastest leads to faulty, i.e. tests corresponding to the conflicts in the set (14b). If the normal status is prioritized, then evaluate tests that correspond to the conflicts in (14a).

## 9 CONCLUSIONS

Motivated by applications used in automotive vehicles, the fault status of a component is defined as faulty, suspected, or normal. Also defined is the readiness of the fault status that states if the evaluation of additional diagnostic tests could change the fault status or not. An important aspect of the readiness defined in this paper is that a component could be ready even though all tests that supervise the component have not been evaluated. The relations between fault status,

readiness and diagnostic tests have been given for both centralized and distributed systems. Using the relations for distributed systems, the fault statuses computed locally can be used to directly compute the global fault status.

The designed FSR algorithm (fault status and readiness) computes the fault status and the readiness for all components. The algorithm gives for example a reduction in processor load with over 80 % for a selective catalytic reduction system (SCR) used in a heavy duty vehicle including over 50 components and 70 diagnostic tests, compared to a direct computation. The fault status and the readiness can also be computed for the automotive application when it is intractable to compute the set of minimal diagnoses, the reason for this is that the complexity only grows linearly in the number of tests for the FSR algorithm while it grows exponentially for the computation of the diagnosis.

After the fault status and the readiness have been computed for all components, it is interesting to know which diagnostic tests that could be evaluated to improve the fault status and the readiness. Conditions have been stated that could be used to exactly compute which diagnostic tests that are meaningful for different components. As an example, if the SCR is affected by two abnormal components then two tests are meaningful. Given a set of meaningful tests, a strategy was designed that computes which meaningful test that will give the most number of ready components. For the SCR affected by the two abnormal components, the best test will in mean give 1.7 new ready components, while the other test will in mean only give 1.0 ready components. By evaluating the meaningful tests in the best order, the number of tests that has to be evaluated is reduced to a minimum, and thereby for example reduces the usage of computational resources.

# APPENDIX

## A THE PROBABILITY FOR A TEST TO RESPOND

In Section 8.1 the conditional probability $P(m|T_R \wedge T_{\neg R})$ for test $m$ to respond given a set of respond tests $T_R$ and a set of non-responded tests $T_{\neg R}$ was used. It was stated that the probability could be computed but that at least the set of minimal cardinality diagnoses is needed. This appendix will show why this is the case. The computations below will consider the ideal case with ideal tests and equal, independent, and low a-priori probability for a component to be abnormal.

The wanted probability can be written as

$$P(m|T_R \wedge T_{\neg R}) = \sum_{X \subseteq \mathcal{C}} P(m|X)P(X|T_R \wedge T_{\neg R}) \tag{16}$$

where the sum is over all combinations of system fault modes, and $X$ is short hand for the system mode

$$\bigwedge_{c \in X} AB(c) \bigwedge_{c \in \mathcal{C} \setminus X} \neg AB(c).$$

Each term in the sum can be expanded to

$$P(m|X)P(X|T_R \wedge T_{\neg R}) = P(m|X)\frac{P(T_R \wedge T_{\neg R}|X)P(X)}{P(T_R \wedge T_{\neg R})}$$

using Bayes rule.

Now, three observations. First: assuming ideal tests, i.e. no missed detection and no false alarms, then the probability

$$P(T_R \wedge T_{\neg R}|X) = \begin{cases} 1 & \text{if } X \in \mathcal{D}^{all} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\mathcal{D}^{all}$ is the complete set of diagnoses. This means that if a test supervises a component that is abnormal then the test has responded, and otherwise it has not responded. Due to this probability, the sum in (16) only have to be performed over the set of diagnoses.

Second: assuming equal, independent, and low a-priori probabilities for abnormal components $P(AB(c))$, the probability

$$P(X \notin \mathcal{D}^{mc}) \ll P(X \in \mathcal{D}^{mc})$$

since $P(X) = P(AB(c))^{|X|}$ and where $\mathcal{D}^{mc}$ is the set of minimal cardinality diagnoses. Therefore, the sum in (16) can be approximated with

a sum over only the set of diagnoses with minimal cardinality. This approximation is accurate as long as the number of non minimal cardinality diagnoses times the probability for an abnormal component is insignificant.

Third: Using (17), the probability $P(T_R \wedge T_{\neg R})$ can be expanded to

$$\begin{aligned} P(T_R \wedge T_{\neg R}) &= \sum_{X \subseteq \mathcal{C}} P(T_R \wedge T_{\neg R}|X)P(X) \approx \sum_{X \in \mathcal{D}^{mc}} P(X) \\ &= |\mathcal{D}^{mc}| \cdot P(AB(c))^{|X^{mc}|} \end{aligned}$$

where $X^{mc}$ is any minimal cardinality diagnosis.

Using the three observations above, the probability (16) can be simplified to

$$\begin{aligned} P(m|T_R \wedge T_{\neg R}) &\approx \sum_{X \in \mathcal{D}^{mc}} P(m|X) \frac{1 \cdot |P(AB(c))^{|X|}}{|\mathcal{D}^{mc}| \cdot P(AB(c))^{|X|}} \\ &= \sum_{X \in \mathcal{D}^{mc}} P(m|X) \frac{1}{|\mathcal{D}^{mc}|}. \end{aligned}$$

To further simplify (16) it can be used that the probability $P(m|X)$ that the test $m$ responds given the abnormal components

$$P(m|X) = \begin{cases} 1 & \text{if } X \in \mathcal{D}^{all} \\ 0 & \text{otherwise} \end{cases}$$

if the test $m$ is assumed to be ideal. The probability (16) can therefore be simplified to

$$P(m|T_R \wedge T_{\neg R}) = \frac{|\{X \in \mathcal{D}^{mc}|\pi(m) \cap X \neq 0\}|}{|\mathcal{D}^{mc}|}. \tag{18}$$

In can be seen in (18) that to compute the probability $P(m|T_R \wedge T_{\neg R})$, the set of minimal cardinality diagnoses is needed. If ideal tests and equal, independent, and small probabilities for abnormal components are not assumed, then the computations become much more difficult and will require that the complete set of diagnoses has been computed.

# *Paper IV*

# SAFETY ANALYSIS OF AUTONOMOUS SYSTEMS BY EXTENDED FAULT TREE ANALYSIS[1]

**Jan Åslund, Jonas Biteus, Erik Frisk,**
**Mattias Krysander, and Lars Nielsen**

*Dep. of Electrical Engineering, Linköpings universitet,*
SE-*581 83 Linköping, Sweden*
{jaasl,biteus,frisk,matkr,lars}@isy.liu.se

## ABSTRACT

Safety is of major concern in many autonomous functions in automotive systems and aerospace. In these application areas it is standard to use fault trees, and a natural question in many modern systems that include sub-systems like diagnosis, fault-tolerant control, and autonomous functions, is how to include the performance of these algorithms in a fault tree analysis for safety. Many possibilities exist but here a systematic way is proposed. It is shown both how safety can be analyzed and how the interplay between algorithm design in terms of missed detection rate and false alarm rate is included in the fault tree analysis. Examples illustrate analysis of diagnosis system requirement specification and algorithm tuning.

[1] This paper has been published as (Åslund et al., 2006).

# 1 INTRODUCTION

Safety is of major concern in many applications, and the interest in safety analysis is increasing (Villemeur, 1992), both in general and for autonomous systems. A key mechanism in an autonomous system is situation classification followed by a decision, which means that decisions that previously were taken by a pilot or a driver, are now taken autonomously. It will thus be necessary to assure safety levels of autonomous vehicles, for example to obtain certification of airworthiness for unmanned aerial vehicles. Such safety analysis is the topic of this paper.

In the safety analysis, both diagnosis (Gertler, 1998; Patton et al., 2000) and fault-tolerant control (Blanke et al., 2003) must be considered and this has created a new set of problem formulations to study. One fundamental question is of course if a system becomes safer when a diagnosis function and fault-tolerant control are introduced, and if so, by how much? Another question is how to formulate specification requirements on diagnosis algorithms so that overall system safety is as good as possible. This also naturally leads to the question of how to select internal design parameters in the diagnosis algorithms. One simple example is that a selection of a threshold balances the rates of missed detection and false alarm. Where to put this balance very much depends on the situation and how it propagates to overall system safety.

To get a handle on these questions it is necessary to have a quantitative method and in this respect fault tree analysis is a natural starting point. It is the basic tool in safety analysis, and may even be a requirement from government, e.g. when declaring air worthiness for aircrafts. Having made this choice, the question is now how to include properties of diagnosis algorithms in fault tree analysis. It should be noted that the main concern here is the interplay between safety and algorithms, and that this should not be confused with the more studied problem on safety of software. It is here assumed that the software is a correct coding of the specified algorithm following the procedures for implementation of safety critical systems.

The purpose of this paper is to put forward the problems described above and to present a solution. In Section 2 fault tree analysis is recapitulated, and in Section 3 diagnosis performance and central concepts like false alarm and missed detection are recalled. It is clear that a fault tree in general, for a certain system, can be formulated in different ways. Nevertheless, in spite of the possible ambiguity in original fault tree formulation, one can look for systematic ways of introducing diagnosis properties in a given fault tree. This is proposed in Section 4 based on algorithm performance in terms of probabilities for false alarm and missed detection. Further, the use of false alarm rate

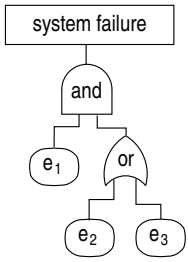



FIGURE 1: A fault tree for the system in Example 1.

and missed detection rate is the link to parameter setting of the algorithms, and thus the foundation for both requirements specification on one hand and for algorithm tuning on the other hand. In Section 5 the proposed methods are applied to these generic examples chosen to illustrate the fundamental questions posed in the beginning of this introduction. Finally, the conclusions are drawn in Section 6.

## 2 FAULT TREE ANALYSIS

Fault tree analysis is a systematic way to investigate credible causes for an undesired event in a system (Stamatelatos and Vesley, 2002; Vesley et al., 1981). The logical relationships between the undesired event and basic events that lead to the undesired event are presented in a fault tree.

In general there might exist dependencies between the basic events and also the same basic event might appear more than once in the fault tree, as will be seen in the examples later on. It is straightforward to treat these dependencies in the calculations and there exists a multitude of fault tree software for this purpose. Thus, if the probabilities for the basic events are known, the expression for the probability of the top event can be computed (Villemeur, 1992).

EXAMPLE 1: Figure 1 shows a fault tree that gives the relationship between the top event *system failure* and the basic events $e_1$, $e_2$, and $e_3$ which are assumed to be independent. The gate symbols denote the relationships between the input events below the gates and the output event above. This fault tree will be used in the following sections where it is assumed that the basic events are sensor failures, i.e. event $e_i$ denotes that sensor $i$ is broken, and the probability for this is denoted by $p_i$. The analytical expression for the probability of the top event can easily be computed as

$$P(\text{system failure}) = p_1(p_2 + p_3 - p_2p_3). \qquad \diamond$$

# 3 DIAGNOSIS PERFORMANCE

A key component in a fault-tolerant control system is a diagnosis system and a common way to perform diagnosis is to use a set of tests. Each of these tests consists of a test quantity $T_i$, and a threshold $J_i$.

The test quantity $T_i$, also called *residual*, is designed such that $T_i$ is small if the system to be diagnosed is OK and large otherwise. The test quantity $T_i$ is compared to a threshold $J_i$ and if $T_i > J_i$ then the test is said to alarm. The decision is that the process to be diagnosed is not okay, i.e. that component $i$ is $\neg$OK. In statistical theory (Berger, 1985) the hypothesis "component $i$ is OK" is called the *null hypothesis* of a test and is denoted $H_i^0$. In (Nyberg, 2002) this statistical theory is included in a diagnosis framework.

To alarm when the supervised system is OK, i.e. $H_i^0$ is true, is called a *false alarm* ($FA_i$). Further, to not alarm when the supervised system is faulty, i.e. $H_i^0$ is false, is called a *missed detection* ($MD_i$). The probability of these two events define important performance measures of a test as follows. The false alarm probability is

$$P_{FA_i} = P(T_i > J_i | H_i^0 \text{ true}) \tag{1}$$

and the missed detection probability is

$$P_{MD_i} = P(T_i \leqslant J_i | H_i^0 \text{ false}). \tag{2}$$

An ideal test gives no false alarms and no missed detections.

When designing tests, the last step is to compute a threshold. Dropping the index $i$ for now, the FA and MD probabilities are, as can be seen in (1) and (2), functions of the threshold $J$. A typical example of FA and MD probabilities as functions of the size of the threshold can be seen in Figure 2. To obtain a small FA probability the threshold must be large, but with a large threshold the MD probability gets large. Hence the choice of threshold adjusts the compromise between a small FA probability and a small MD probability.

# 4 INCLUDING DIAGNOSIS PERFORMANCE AND AUTONOMOUS DECISIONS IN A FAULT TREE

By including the effects of diagnosis algorithms and fault-tolerant control in fault trees, the consequences of false alarms and missed detections are explicitly modeled. It will later be shown that for example the compromise between FA and MD can be handled by using the tree including diagnosis and fault-tolerant control. It is described how one

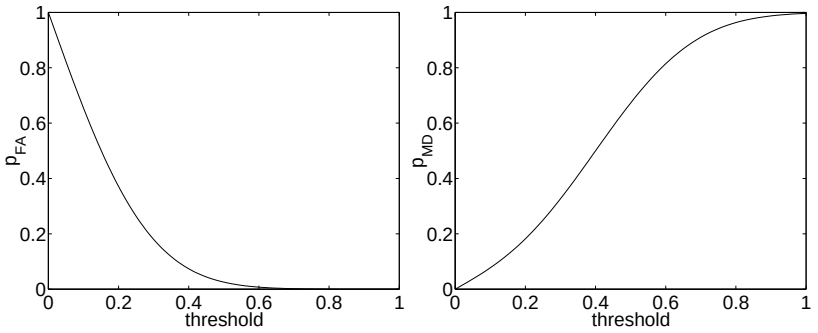



FIGURE 2: Probabilities of FA and MD as functions of threshold size. for a test.

test can be included in a fault tree and a generalization to several tests will be straightforward. Before a systematic method to include diagnosis and fault-tolerant control in an existing fault tree will be presented, an illustrative example is given.

EXAMPLE 2: Consider Example 1 and its fault tree shown in Figure 1. In order to decrease the system failure probability, a backup system, a diagnosis test, and fault-tolerant control are added to the original system. With these new parts, the system can now autonomously adapt to possibly faulty situations by switching to the backup system in case of an alarm. Expressed in logic this means that, the system is OK if the original system is OK or the backup system is switched on and is OK. To be mapped to the fault tree, an equivalent statement is used: the system is not OK if the original system is not OK and the backup system is switched off or if it is switched on and not OK, which is the main structure of Figure 3. The backup system consists of a sensor called sensor 4. The backup system is not OK if sensor 4 is *not* OK, and this event is denoted $e_4$. The test supervises if sensor 2 is OK, i.e. $\neg e_2$.

The expanded fault tree where diagnosis and backup system are included is shown in Figure 3. The tree in Figure 1 can be found in the left branch of the expanded tree. Original system failure is connected to an and-gate together with no alarm, because original system failure leads only to a system failure in absence of an alarm, i.e. the alarm deactivated tree. The right part of the tree describes the logic when alarming, i.e. the alarm-activated tree. This branch consists of a backup failure tree and the alarm tree $A$. The important diagnosis events FA and MD are leafs in this tree. ◇

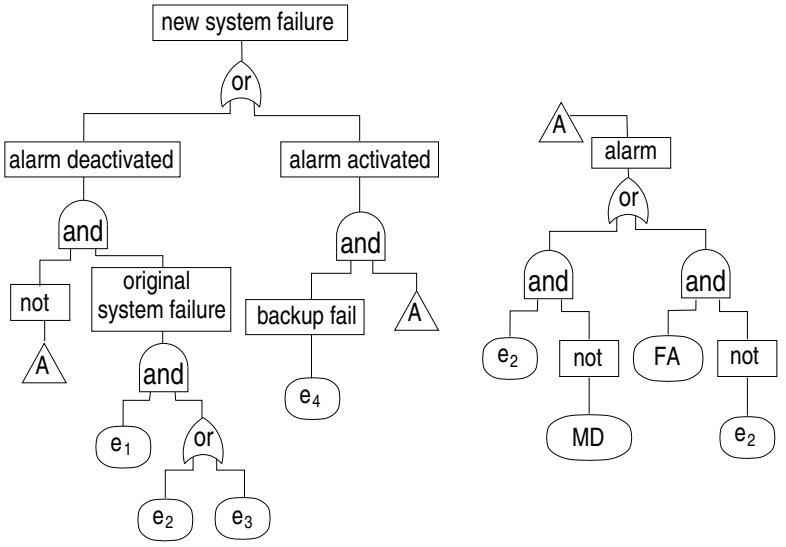



FIGURE 3: A fault tree where diagnosis performance is included.

## 4.1 *A Systematic Approach*

Now a general and systematic inclusion of diagnosis will be described. The fault tree describing the original system without a diagnosis system will be denoted by $T_0$. The procedure consists of four steps, which are described in Section 4.1 to 4.1. The four steps are briefly described as 1) construct an alarm tree, 2) construct an alarm-activated tree, 3) insert the alarm tree in the original tree, and 4) insert the alarm-activated tree in the tree obtained in step 3. In general, there can be several alarm-deactivated trees and alarm-activated trees for one test. The following procedure only describes the case with one alarm deactivated tree and one alarm-activated tree. In the case of several such trees, some of the steps have to be performed several times.

**The alarm tree**

The first step of the inclusion of a test is as said before to construct an alarm tree that describes the event *alarm*. To include the diagnosis performance measures (1) and (2), the alarm event has to be expressed using FA and MD. An alarm can either be a false alarm if $H^0$ is true or a correct alarm if $H^0$ is false. By the definition of MD it follows that a correct alarm is equivalent to the event *not missed detection*. Hence, alarm for a specific test can always be expressed as

$$\text{alarm} = (\text{FA} \wedge \text{H}^0) \vee (\neg\text{MD} \wedge \neg\text{H}^0). \tag{3}$$

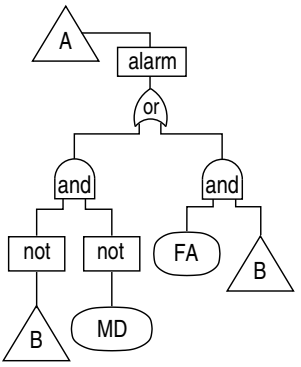
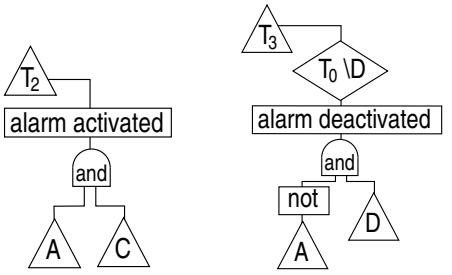



FIGURE 4: A general fault tree describing the alarm event. Everything except for the tree B is fixed.




(a) Step 2. (b) Step 3.

FIGURE 5: Trees constructed in the different steps.

A general way to express this alarm event as a fault tree is shown in Figure 4. By using this general form of an alarm tree, the remaining task is to model the tree denoted by B describing when the null hypothesis $H^0$ is true. This is done by using the description of the test.

To illustrate the first step for Example 2, the null hypothesis $H^0$ of the test is that sensor 2 is OK, i.e. $\neg e_2$. Therefore, if B is substituted in Figure 4 by $\neg e_2$, the alarm tree to the right in Figure 3 is obtained.

### The alarm-activated tree

The second step is to construct the alarm-activated tree $T_2$. The purpose is to model events that are added to the fault tree in case of an alarm. In Example 2 and Figure 3 this was the right part of the left tree. The general structure is shown in Figure 5(a). The tree $T_2$ consists of the alarm tree A obtained in the first step and a sub-tree denoted C.

The sub-tree C is to be constructed such that it describes events that only affect the failure probability when an alarm has occurred.

In Example 2 and Figure 3, sub-tree C corresponds to the sub-tree describing backup failure. Since the backup system is turned on only during an alarm, it can of course only affect the failure probability during an alarm.

**The alarm deactivated tree**

The third step is to construct the alarm deactivated tree $T_3$. The purpose is to describe which events that does not contribute to the top event in case of an alarm. In Example 2 and Figure 3 this was the left part of the left tree, i.e. the deactivation of the suspected faulty components. The general structure is shown in Figure 5(b). The tree $T_3$ consists of a sub-tree D of $T_0$, which was the original tree, the alarm tree A constructed in step 1, and $T_0 \backslash D$, where $T_0 \backslash D$ denotes the sub-tree in $T_0$ induced by the vertices not in D. This means that $T_3$ is uniquely determined if D is known. The tree D is defined as the sub-tree of the original fault tree $T_0$ that affects the failure probability only and just only when no alarm has occurred, i.e. it should not give any effect when the alarm is active.

In Example 2, the original fault tree is shown in Figure 1. Since the entire original system is turned off in the occurrence of an alarm, it follows that the entire original tree is equal to D. This means that $T_0 \backslash D$ in this example is empty and the resulting tree $T_3$ is the left branch of the tree in Figure 3. Even though $e_2$ is included in D, it can affect system failure through the alarm tree when alarming. However, since this dependence is explicitly handled in A, it should not be considered when identifying D.

**Final complete tree**

The fourth and final step will give the desired fault tree including the original fault tree and the diagnosis system. This means that the tree $T_2$ is attached to the tree $T_3$, which results in the final fault tree. Depending on how the diagnosis system is included, tree $T_2$ is either included with an OR gate or with an AND gate.

For Example 2, the backup system during an alarm has the same function as the original system during no alarm. From the system description, it follows that the tree $T_2$ are inserted with an OR gate directly below the first AND gate, resulting in the final tree shown in Figure 3.

EXAMPLE 3: Consider again Example 1, but in a new scenario. Let sensor 1 be supervised. If the test alarms, a prediction of the measured value of sensor 1 is automatically used instead. The prediction

is based on an observer that uses measurements of sensor 2. The resulting fault tree is shown in Figure 6 where the four trees A, B, C, and D are encircled. How to obtain this result will next be explained by following the proposed procedure step by step.

Since the test checks if sensor 1 is OK, the null hypothesis $H^0$ of the test is $\neg e_1$, which defines sub-tree B. The alarm tree A is straightforwardly constructed and step 1 is completed. When an alarm occurs, a predicted value of the measurement is used. Therefore, the sub-tree C is in this example describing the event *bad prediction*. A bad prediction is a consequence of $e_2$ or a faulty observer algorithm denoted by *observer faulty* in Figure 6. The resulting alarm-activated tree $T_2$ is the *active bad prediction*-tree.

To perform step 3, notice that sensor 1 is turned off during an alarm. Hence it is only $e_1$ in the original tree that does not affect the system failure during an alarm, i.e. $D = e_1$. The alarm deactivated event is denoted by *active bad value* in Figure 6. In step 4, recall that sensor 1 is predicted in case of an alarm, i.e. a precaution is taken to reduce the risk of *active bad value* of sensor 1, which is the event $e$ for this example. By introducing the event *bad value* combining the obtained trees as the system description specify, the fault tree in Figure 6 is obtained. ◇

## 4.2 *Simplifications of the Fault Tree by Using Approximations*

It is sometimes possible to use approximations to obtain a simpler fault tree. One objective is to introduce diagnosis performance closer to the supervised event, which is desirable if the fault tree is large and difficult to survey.

To illustrate this idea, the fault tree in Figure 3 is used. The top event in this tree is system failure (SF). Consider for example a scenario where alarms are rare, i.e.

$$P(\neg\text{alarm}) \approx 1. \tag{4}$$

Now it will be shown how this approximation can be used to simplify the fault tree. Since alarm and ¬alarm are mutually exclusive events whose probabilities sum to one, it holds that

$$P(SF) = P(SF|\neg\text{alarm})P(\neg\text{alarm}) + P(SF|\text{alarm})P(\text{alarm}). \tag{5}$$

Consider the first term in the right-hand side. Here approximation (4) can be used and the conditional probability $P(SF|\neg\text{alarm})$ can be calculated using the fault tree for original system in Figure 1, with the original probabilities $P(e_i)$, $i = 1, 2, 3$ replaced by conditional probabilities $P(e_i|\neg\text{alarm})$. The probabilities for the events $e_1$ and $e_3$ are not

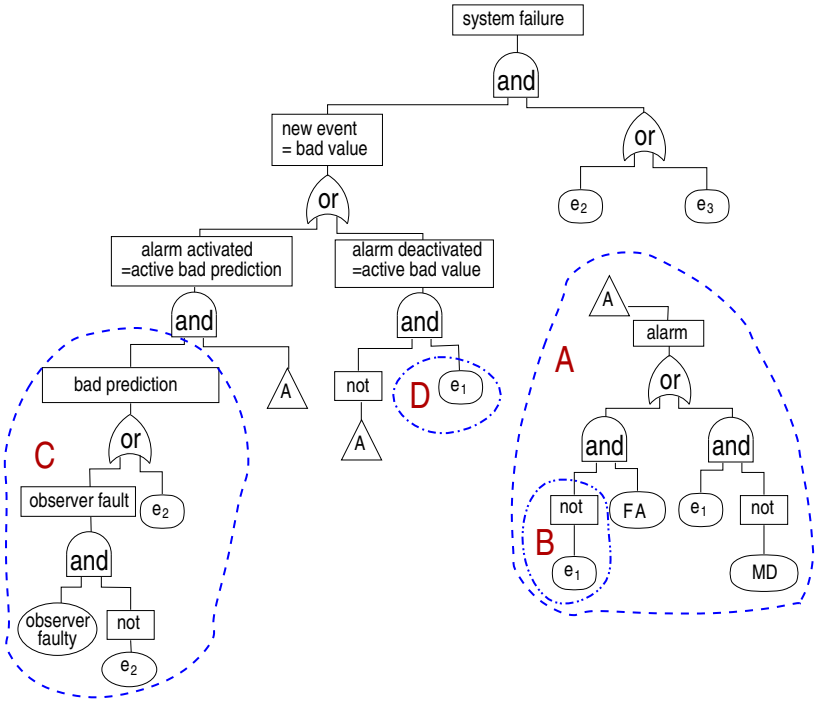



FIGURE 6: Tree including diagnosis performance.

affected by the conditioning since they are independent of the supervised event $e_2$, and using approximation (4) and that $P(\neg alarm|e_2) = P_{MD}$ the following approximation of the conditional probability for event $e_2$ is obtained

$$P(e_2|\neg alarm) = \frac{P(\neg alarm|e_2)P(e_2)}{P(\neg alarm)} \approx P_{MD}P(e_2). \tag{6}$$

The approximations of the first term in (5) are represented by the left branch of the fault tree in Figure 7. The second term in (5) corresponds to the right branch in the fault tree and is the same as before.

One objective with the approximation was to introduce the diagnosis performance closer to the supervised event. In Figure 7 the event MD is next to the supervised event $e_2$. Thus, compared to the tree in Figure 3, a more local representation has been obtained. This effect becomes more pronounced in larger examples.

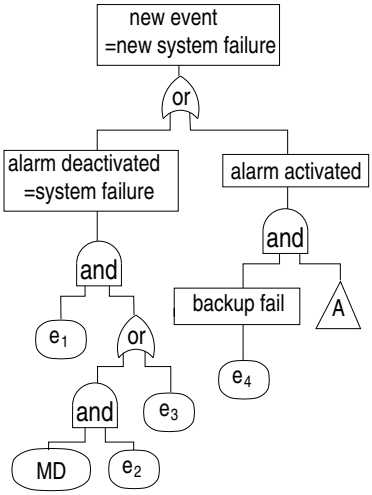



FIGURE 7: A simplification of the fault tree in Figure 3.

# 5 GENERIC ILLUSTRATIVE EXAMPLES

Previous sections have described how to specify and incorporate performance of diagnosis algorithms in fault trees for overall system safety. The main points of the paper is that systematic inclusion of diagnosis in fault trees can be used in both design and requirement specification of diagnosis algorithms. Here, generic examples are presented that shows how this can be done.

## *5.1 Performance Requirements on the Diagnosis Algorithm*

Suppose that external requirements state numerical demands on the probability for the top event, e.g. the probability for a system failure, so that the demand is

$$P(\text{top event}) \leqslant \beta$$

where $\beta$ is the performance specification. If it has been concluded that this requirement can not be fulfilled without a diagnosis system, then a question is what performance requirements on the diagnosis system are necessary to ensure that the overall requirement is fulfilled.

As described in Section 3, the performance of a diagnosis test can be condensed into the probability for false alarm, $P_{FA}$, and missed detection, $P_{MD}$. In the analysis in this section, no specific algorithm is considered, and therefore the performance of an algorithm is here specified in a diagram with $P_{MD}$ and $P_{FA}$ on the axes. As an example, Figure 8 shows the performance of the diagnosis algorithm from

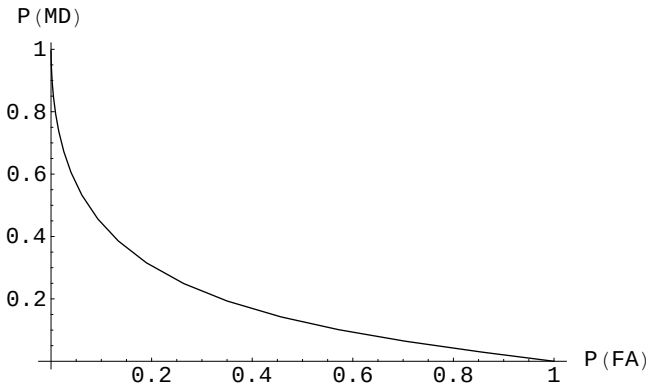



FIGURE 8: The curve indicates performance of a diagnosis algorithm in a $P_{FA}/P_{MD}$ diagram.

Section 3. The curve is parameterized by the threshold and is directly obtained from Figure 2.

Introducing the diagnosis system in the fault tree, the probability of the top event becomes a function of the performance measures $P_{MD}$ and $P_{FA}$ in addition to all the probabilities for the basic events in the original fault tree, $p_1, \ldots, p_N$. Since the probabilities $p_i$ are fixed, the probability for the top event becomes a function of only $P_{MD}$ and $P_{FA}$, i.e.

$$P(\text{top event}) = f(P_{FA}, P_{MD}). \tag{7}$$

The expression for $f$ can for example be computed based on the fault tree, see Section 2. The overall performance requirement can now be stated as

$$f(P_{MD}, P_{FA}) \leqslant \beta. \tag{8}$$

The requirement specification on the diagnosis algorithm will then be the set of $(P_{FA}, P_{MD})$ that satisfies (8).

EXAMPLE 4: Consider again Example 2, where sensor 2 is supervised. In this example, the probabilities for the basic events are assumed to be $p_1 = 0.1$, $p_2 = 0.005$, $p_3 = 0.01$, $p_4 = 0.005$, where $p_i = P(e_i)$. Assume that the overall requirement is that the probability for the top event must be lower than $\beta = 0.00145$. Based on only this, it is not clear if this performance is at all possible, and in case it is possible, which trade-off between false-alarms and detection performance is needed?

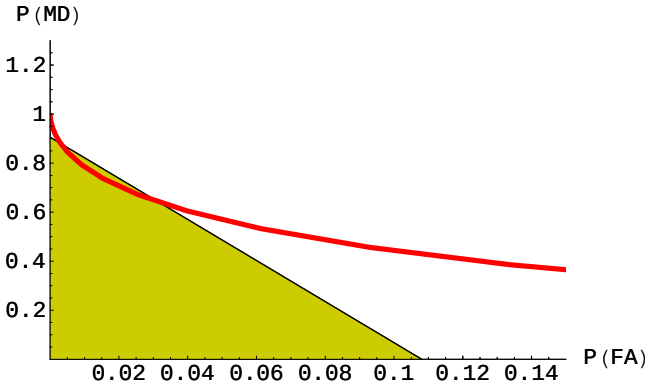



FIGURE 9: Feasible diagnosis performance (shaded) and performance of the detection algorithm (curve).

Performing the analysis outlined above results in the requirement specification on the diagnosis algorithm which is indicated by the shaded area in Figure 9. In this case, inequality (8) becomes

$$(9)\quad (1-p_2)(p_4-p_1p_3)P_{FA}+p_2(p_1-p_4)P_{MD}+p_1(1-p_2)p_3+p_2p_4 \leqslant \beta.$$

which is linear in $P_{MD}$ and $P_{FA}$. The solid line shows the performance of the diagnosis algorithm from Section 3, and since the intersection is non-empty (approximately for $0.005 \leqslant P_{FA} \leqslant 0.03$), there exist a feasible tuning, i.e. threshold selection, of the algorithm that fulfills overall system requirements. It is clear from the figure that to meet the safety performance specification, using the diagnosis system from Example 2, we need a low probability for false-alarm at the expense of a relatively high probability for missed detection. This trade-off was not immediate from the problem formulation and could only be obtained by analyzing the effect of the diagnosis system on the overall safety. ◇

The function $f$ in (8) is a low order polynomial in the probabilities $P_{FA}$ and $P_{MD}$ and even linear in the case of only one diagnosis algorithm. The linearity property also holds for the case of several independent diagnosis algorithms. For cases with more than one diagnosis algorithm that are connected in the fault tree, cross-terms appear in $f$ and the maximum order of the polynomial equals the number of diagnosis tests.

In practice, there are often several top events where each event is represented using a fault tree. Often, these events lead to opposing requirements, e.g. systems safety vs. availability, and there is a need to

make a compromise. Extension to the case with more than one top event is straightforward. Apply the procedure above for each tree to obtain a set of feasible performance specifications. Then validate that the intersection of all these sets is non-zero, i.e. that there exist false alarm/missed detection probabilities that satisfies all top event requirements.

## 5.2 *Optimal Threshold Selection*

In the section above it was shown how the fault tree could be used for determining required performance of a diagnosis algorithm such that overall safety specifications are met. This section will show that, if a specific diagnosis algorithm is selected, the fault tree can also be used for optimal algorithm tuning, for example threshold selection.

As noted in (7), the probability $P(\text{top event})$ can be written as a function of $P_{MD}$ and $P_{FA}$. Given a diagnosis algorithm, the probabilities $P_{MD}$ and $P_{FA}$ are in their turn functions of the threshold $J$, as described in Section 3. The probability for the top event can therefore be written as

$$P(\text{top event}) = f(P_{MD}(J), P_{FA}(J)) = F(J).$$

In Figure 10 the function $F$ is shown for the fault tree in Figure 3 with the diagnosis performance measures from Figure 2. In this example it is natural to choose $J$ so that $P(\text{top event})$ is as small as possible, i.e. to solve the optimization problem

$$\min_{J \in I} F(J)$$

where $I$ is the set of admissible values for $J$. In the figure, it is clear that the optimal choice of $J$ is around $J = 0.5$ which, according to Figure 2, corresponds to a low false-alarm probability and a high missed detection probability. The case with several fault trees with different top events leads to a multi-objective optimization problem with more than one objective function.

One may note that an analysis like the one above is not only possible for parameters of the diagnosis system. Consider for example the observer in Example 3 that is part of the fault tolerant control system. In the same way as the threshold of the diagnosis algorithm influences the overall safety, so does the observer gain. If the influence of the observer gain, or pole placement, is modeled in the fault tree, it is straightforward to include also these parameters in the optimization.

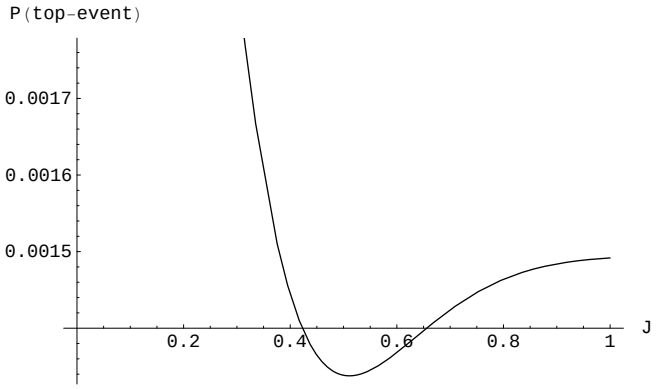



FIGURE 10: Probability for the top event as function of the threshold.

# 6 CONCLUSIONS

Autonomous systems is often a complex combination of systems and in many cases diagnosis and fault tolerant control are integrated to increase availability and safety. To be able to analyze if a system is safe or not, the common approach is to use fault tree analysis. Introducing diagnosis systems and fault-tolerant control in the fault tree makes it possible to include the effects of these systems in the safety analysis.

A systematic way to include diagnosis and fault-tolerant control in fault tree analysis was presented. A fault tree for a given system can be formulated in many ways, but nevertheless Section 4 gives a systematic method that expands the given tree to include also the diagnosis system and the fault-tolerant control algorithms. In Section 5.1 it was shown how requirements on overall system performance can be systematically transferred, via a fault tree analysis, into performance requirements on the diagnosis system. Further, in Section 5.2 it was shown how an optimization criterion in a similar straightforward way is obtained, making optimization of parameter tuning simple.

In conclusion, a major advantage of the proposed methodology is that it is structured so that it enables tools for interaction regarding the interplay between algorithms and safety, thus resulting in better systems and savings in valuable engineering time.

# REFERENCES


Amazon, 2007. Amazon, Jan 2007. `http://www.amazon.com/`.

J. Åslund, J. Biteus, E. Frisk, M. Krysander, and L. Nielsen. A systematic inclusion of diagnosis performance in fault tree analysis. In IFAC *World Congress*, Praha, Czech Republic, 2005.

J. Åslund, J. Biteus, E. Frisk, M. Krysander, and L. Nielsen. Safety analysis of autonomous systems by extended fault tree analysis. *International Journal of Adaptive Control and Signal Processing*, 21(2–3): 287–298, Oct 2006.

AA1CAR, 2007. AA1CAR Automotive Repair and Diagnostic Help, Jan 2007. `http://www.aa1car.com/`.

M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes*. PTR Prentice-Hall, Inc, 1993.

J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 1985.

J. Biteus. Distributed diagnosis and simulation based residual generators. Technical report, Dept. of Electrical Engineering, 2005. LiU-TEK-LIC-2005:31, Thesis No. 1176.

J. Biteus and M. Nyberg. Dynamic evaluation of minimal structurally singular sets. In CCSSE, Norrköping, Sweden, 2002.

J. Biteus and M. Nyberg. Residual generators for DAE systems utilizing minimal subsets of model equations. In *5th* IFAC *Symposium on Fault Detection, Supervision and Safety of Technical Processes, safeprocess*, Washington, DC, USA, June 2003. IFAC, Elsevier Science Ltd., Oxford, UK.

J. Biteus, G. Cedersund, E. Frisk, M. Krysander, and L. Nielsen. Improving airplane safety by incorporating diagnosis into existing safety practice. Technical Report LiTH-ISY-R-2648, Dept. of Electrical Engineering, Linköpings universitet, Sweden, 2004a.

J. Biteus, M. Jensen, and M. Nyberg. Decentralized diagnosis in heavy duty vehicles. In *Fourth Conference on Computer Science and Systems Engineering Linköping*, Norrköping, Sweden, Oct 2004b.

J. Biteus, M. Jensen, and M. Nyberg. Distributed diagnosis for embedded systems in automotive vehicles. In IFAC *World Congress*, Praha, Czech Republic, 2005.

J. Biteus, E. Frisk, and M. Nyberg. Condensed representation of global diagnoses with minimal cardinality in local diagnoses. In *17th International Workshop on Principles of Diagnosis* DX'*06*, Spain, 2006a.

J. Biteus, M. Nyberg, E. Frisk, and J. Åslund. Determining a component's fault status and the status' readiness. In *Proceedings of* IFAC *Safeprocess'06*, Beijing, China, 2006b.

J. Biteus, M. Nyberg, and E. Frisk. An algorithm for computing the diagnoses with minimal cardinality in a distributed system. IFAC, *Engineering Applications of Artificial Intelligence*, 2007. Accepted for publication.

M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 2003.

Bosch, 2006. Bosch internet press center, Oct 2006. `http://www.bosch.com/`.

J. Chen and R. J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers, 1999.

M.-O. Cordier, P. Dague, F. Levy, J. Montmain, M. Staroswiecki, and L. Trave-Massuyes. Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *Systems, Man and Cybernetics, Part B,* IEEE *Transactions on*, 34(5):2163–2177, Oct 2004.

R. Debouk, S. Lafortune, and D. Teneketzis. Discrete event dynamic systems. *Coordinated Decentralized Protocols for Failure Diagnosis of Discrete Event Systems*, 10:33–86, Jan 2000.

O. Dressler and P. Struss. *Principles of Knowledge Representation*, chapter The consistency-based approach to the automated diagnosis of devices. CSLI Publications, 1996.

EPA, 2005. *On-board diagnostics (OBD)*. US Environment Protection Agency (EPA), 2005. `http://www.epa.gov/`.

EU, 2005. *Emissions of atmospheric pollutants from motor vehicles*. European Union, 2005. `http://www.europa.eu.int/`.

G. Friedrich, G. Gottlob, and W. Nejdl. Physical impossibility instead of fault models. In *Proceedings of 8th National Conf. on Artificial intelligence*, Boston, USA., 1990.

E. Frisk. *Residual Generation for Fault Diagnosis*. PhD thesis, Linköpings Universitet, November 2001.

J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, 1998.

J. Gertler, M. Costin, X. Fang, Z. Kowalczuk, M. Kunwer, and R. Monajemy. Model based diagnosis for automotive engines-algorithm development and testing on a production vehicle. *Control Systems Technology,* IEEE *Transactions on*, 3(1):61–69, Mar 1995.

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

F. Harary. *Graph Theory*. Addison-Wesley Publishing Co., Boston, USA, 1969.

C. C. Hayes. Agents in a nutshell-a very brief introduction. *Knowledge and Data Engineering,* IEEE *Transactions on*, 11(1):127–132, Jan/Feb 1999.

D. Hristu-Varsakelis and W. S. Levine. *The Handbook of Networked and Embedded Control Systems*. Springer Verlag, 2005.

R. Isermann. Diagnosis methods for electronic controlled vehicles. *Vehicle System Dynamics*, 36(2–3):77–117, Sep 2001.

R. Isermann. Model-based fault-detection and diagnosis - status and applications. *Annual Reviews in Control*, 29(1):71–85, 2005.

ISO, 1999. *ISO 14230: Road vehicles – Diagnostic systems – Keyword Protocol 2000*. International Organization for Standardization, 1999.

F. V. Jensen and T. Nielsen. *Bayesian networks and Decision Graphs*. Springer Verlag, 2001.

J. de Kleer. Focusing on probable diagnoses. In AAAI, pages 842–848, 1991.

J. de Kleer and J. Kurien. Fundamentals of model-based diagnosis. In *Proceedings of* IFAC *Safeprocess'03*, Washington, USA, 2003.

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

J. de Kleer and B. Williams. Diagnosis with behavioral modes. In IJCAI-*89*, pages 104–109, Detroit, USA, 1989.

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

M. Krysander. *Design and Analysis of Diagnosis Systems Using Structural Methods*. PhD thesis, Linköpings universitet, June 2006.

J. Kurien, X. Koutsoukos, and F. Zhao. Distributed diagnosis of networked, embedded systems. In *Thirteenth International Workshop on Principles of Diagnosis*, Semmering, Austria, May 2002.

G. Leen and D. Heffernan. Expanding automotive electronic systems. *Computer*, 35(1):88–93, Jan 2002.

J. Lunze and J. Schröder. State observation and diagnosis of discrete-event systems described by stochastic automata. *Discrete Event Dynamic Systems*, 11(4):319–369, Oct 2001.

N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert. Trends in automotive communication systems. *Proceedings of the* IEEE, 93(6):1204–23, Jun 2005.

M. Nyberg. *Model Based Fault Diagnosis: Methods, Theory, and Automotive Engine Applications*. PhD thesis, Linköpings Universitet, June 1999a.

M. Nyberg. Automatic design of diagnosis systems with application to an automotive engine. *Control Engineering Practice*, 7(8):993–1005, August 1999b.

M. Nyberg. Model-based diagnosis of an automotive engine using several types of fault models. *Control Systems Technology,* IEEE *Transactions on*, 10(5):679–689, September 2002.

R. Patton, P. Frank, and R. Clark, editors. *Issues of Fault Diagnosis for Dynamic Systems*. Springer, 2000.

C. Price. *Computer-Based Diagnostic Systems – Computer-Based Troubleshooting*. Springer Verlag, 1999.

G. Provan. A model-based diagnosis framework for distributed systems. In *Proc. of the Thirteenth International Workshop on Principles of Diagnosis*, Semmering, Austria, May 2002.

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

N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *2nd Conference on Autonomous Agents and Multi-Agent Systems*, Australia, July 2003.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *Automatic Control,* IEEE *Transactions on*, 40(9), 1995.

Scania, 2007. Scania internet press center, Jan 2007. `http://www.scania.com/`.

J.-Y. Shin and C. M. Belcastro. Performance analysis on fault tolerant control system. *Control Systems Technology,* IEEE *Transactions on*, 14(5): 920–925, Sep 2006.

M. Stamatelatos and W. Vesley. Fault tree handbook with aerospace applications. Technical report, NASA, USA, 2002.

SAE, 2003. *On-Board Diagnostics for Light and Medium Duty Vehicles Standards Manual*. Society of Automotive Engineers, Warrendale, USA, 2003.

P. Struss and C. Price. Model-based systems in the automotive industry. AI *Magazine*, 24(4):17–34, 2003.

R. Su and W. M. Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *Automatic control,* IEEE *transactions on*, 50(12):1923–35, Dec 2005.

A. S. Tanenbaum and M. van Steen. *Distributed Systems*. Prentice Hall, 2002.

L. Trave-Massuyes, T. Escobet, and X. Olive. Diagnosability analysis based on component-supported analytical redundancy relations. *Systems, Man and Cybernetics, Part A,* IEEE *Transactions on*, 36(6):1146–1160, Nov 2006.

S. Tuhrim, J. Reggia, and S. Goodall. An experimental study of criteria for hypothesis plausibility. *Journal of Experimental & Theoretical Artificial Intelligence*, 3(2):129–144, 1991.

W. Vesley, Goldberg, Roberts, and Haasl. Fault tree handbook. Technical Report NUREG-0492, US NRC, USA, 1981.

A. Villemeur. *Reliability, availability, maintainability and safety assessment*, volume 1 & 2. John Wiley & sons, UK, 1992.

G. Weiss, editor. *Multiagent systems : a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, Mass., USA, 1999.

F. Wotawa. A variant of reiter's hitting-set algorithm. *Information Processing Letters*, (79):45–51, 2001.

Notes

Notes

Linköpings Studies in Science and Technology
Department of Electrical Engineering
Dissertions, Vehicular Systems

No. 6 *Design and Analysis of Diagnosis Systems Using Structural Methods*
Mattias Krysander, Dissertation No. 1033, 2006.

No. 5 *Air Charge Estimation in Turbocharged Spark Ignition Engines*
Per Andersson, Dissertation No. 989, 2005.

No. 4 *Residual Generation for Fault Diagnosis*
Erik Frisk, Dissertation No. 716, 2001.

No. 3 *Model Based Fault Diagnosis: Methods, Theory, and Automotive Engine Applications,* Mattias Nyberg, Dissertation No. 591, 1999.

No. 2 *Spark Advance Modeling and Control*
Lars Eriksson, Dissertation No. 580, 1999.

No. 1 *Driveline Modeling and Control*
Magnus Pettersson, Dissertation No. 484, 1997.