Insitute for Software Technology

Roxane Koitz-Hristov, Dipl.-Ing. Dipl.-Ing. BSc

From Theory to Practice: Abductive Model-based Diagnosis and its Industrial Application

DISSERTATION

to achieve the university degree of

Doctor of Technical Sciences

submitted to

Graz University of Technology

Supervisor Univ.-Prof. Dipl.-Ing. Dr. techn. Franz Wotawa
Insitute for Software Technology
Graz University of Technology

Graz, June 2018

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum	Unterschrift

Abstract

Due to the increasing complexity of technical systems, efficient and effective fault diagnosis is crucial in order to reduce service costs and system downtime. Particularly in domains where maintenance poses an extensive part of the entire operation cost, accurate identification of failure sources has a large economic impact. Model-based diagnosis, as a subfield of Artificial Intelligence, derives explanations for observed anomalies by relying on a formal representation of the artifact under consideration. Two notions of model-based diagnosis exist; first, the consistency-based variation determining root causes through discrepancies between the predicted and actual behavior and second, the abductive form which is founded in logic-based abduction. Although the theoretical background was established decades ago and various research prototypes have been implemented, industrial applications are sparse. The barriers separating science from practice include among others the initial modeling effort, the high computational complexity associated with the fault identification task, and the difficulties of consolidating model-based diagnosis tools and existing software.

To bridge the gap between academic research and industrial applications, this thesis proposes a methodology for applying abductive model-based diagnosis to an industrial setting. We define a process that positions the diagnostic task in the overall operational life of a technical system and enables an easy integration into practice by splitting model generation, fault detection, and fault identification into separate modules. To reduce or even eliminate the initial modeling effort, we automatically construct suitable diagnosis models from failure assessments available in practice. Within the troubleshooting portion of the process, we show a general technique for combining diagnosis, solution ranking, and the selection of probing points. An essential aspect when transporting research into practical applications is to ensure the calculation of solutions is done in a reasonable time frame. Hence, we investigate computational approaches to diagnostic problem-solving which allow us to extract explanations efficiently. Thus, based on the characteristics of the models generated on top of the failure records, we explore different notions of abductive reasoning such as conflict-driven or direct procedures. Our experimental evaluations show that for simple bipartite abduction problems deriving explanations via hitting set computation is favorable. Yet, for more expressive representations we could not determine that either conflict-directed or direct methods are superior.

The practicality of the proposed framework for incorporating abductive model-based diagnosis into real-world domains is evaluated in an industrial use case on wind turbines. Turbine maintenance costs are high and idle times lead to substantial revenue loss making exact and fast fault identification a highly relevant pursuit. Therefore, we apply the theoretically formalized diagnosis process by exploiting automatically retrieved turbine health variables and expert knowledge of failures and their effects. To consolidate the fault identification engine with the existing monitoring software and current work processes, we present a user interface and interaction design as well as a workflow that should foster user acceptance of the diagnosis application. The evaluations up to this point have shown that the diagnosis approach is appropriate and permits practical usage.

Kurzfassung

Aufgrund der zunehmenden Komplexität technischer Systeme ist eine effiziente und effektive Fehlerdiagnose entscheidend, um Servicekosten und Systemausfallzeiten niedrig zu halten. Insbesondere in Bereichen, in denen Wartung einen großen Teil der Gesamtbetriebskosten ausmacht, hat die akkurate Identifizierung von Fehlerquellen große wirtschaftliche Auswirkungen. Die modellbasierte Diagnose als Teilgebiet der Künstlichen Intelligenz leitet Erklärungen für beobachtete Anomalien ab, indem sie sich auf eine formale Darstellung des betrachteten Artefakts stützt. Zwei Richtungen der modellbasierten Diagnose können in der Literatur identifiziert werden: erstens, die traditionelle konsistenzbasierte Variante bestimmt Ursachen durch Diskrepanzen zwischen dem vorhergesagten und dem tatsächlichen Verhalten des Systems und zweitens die abduktive Form, die in der logikbasierten Abduktion begründet ist. Obwohl die theoretischen Grundlagen bereits seit Jahrzehnten etabliert sind und bereits mehrere Forschungsprototypen implementiert wurden, sind industrielle Anwendungen spärlich. Zu den Barrieren, die Wissenschaft und Praxis trennen, gehören unter anderem der anfängliche Modellierungsaufwand, die mit der Fehleridentifikationsaufgabe verbundene hohe rechnerische Komplexität und die Schwierigkeiten des Zusammenführens von modellbasierten Diagnosewerkzeugen und bestehender Software.

Um die Lücke zwischen akademischer Forschung und industriellen Anwendungen zu schließen, wird in dieser Arbeit eine Methodik präsentiert, um abduktive modellbasierte Diagnose in einem industriellen Umfeld anzuwenden. Wir definieren einen Prozess, der die Diagnoseaufgabe innerhalb der Gesamtlebensdauer eines technischen Systems darstellt und eine einfache Integration in die Praxis ermöglicht, indem Modellgenerierung, Fehlererkennung und Fehleridentifikation in separate Module aufgeteilt werden. Um den initialen Modellierungsaufwand zu reduzieren oder gar zu eliminieren, leiten wir aus in der Praxis verfügbaren Fehleranalysedokumenten automatisch geeignete Diagnosemodelle ab. Ein wesentlicher Aspekt beim Übertragen von Forschungsergebnissen in die Praxis besteht darin, sicherzustellen, dass die Berechnung der Lösungen in einem angemessenen Zeitrahmen erfolgt. Daher untersuchen wir computergestützte Ansätze zur Lösung von Diagnoseproblemen, die eine effiziente Ableitung von Erklärungen ermöglichen. Basierend auf den Eigenschaften der Modelle, die auf Grundlage der Fehleranalysedokumente erstellt werden, untersuchen wir unterschiedliche Konzepte des abduktiven Schließens, wie beispielsweise konfliktbasierte oder direkte Verfahren. Unsere experimentellen Auswertungen zeigen, dass für einfache bipartite Abduktionsprobleme die Ableitung von Erklärungen durch Hitting Set Berechnung vorteilhaft ist, während wir für aussagekräftigere Darstellungen nicht feststellen konnten, dass konfliktgesteuerte oder direkte Methoden überlegen sind.

Die Anwendbarkeit des vorgeschlagenen Frameworks für die Integrierung abduktiver modellbasierter Diagnose anhand von realen Problemstellungen wird in einem industriellen Anwendungsfall an Windkraftanlagen untersucht. Die Kosten für die Wartung von Turbinen sind hoch und Ausfallzeiten führen zu erheblichen Umsatzverlusten, was eine genaue Fehleridentifizierung zu einem höchst relevanten Problem macht. Wir wenden den theoretisch formalisierten Diagnoseprozess an, bei dem wir automatisch gelieferte Turbinenzustandsvariablen und Expertenwissen über Fehler und ihre Auswirkungen ausnutzen. Um die Diagnose-Berechnung in die vorhandene Überwachungssoftware und den aktuellen Arbeitsprozess einzubinden, präsentieren wir ein Design für die grafische Benutzeroberfläche sowie einen Arbeitsablauf, der die Benutzerakzeptanz der Diagnoseanwendung fördern soll. Die bisherigen Auswertungen haben gezeigt, dass der Diagnoseansatz geeignet ist und eine praktische Nutzung zulässt.

Acknowledgement

> *Praise the bridge that carried you over.*
>
> **— George Colman the Younger**
> "*Heir-at-Law*". 1797.

My sincere thanks go out to my supervisor Franz Wotawa, who has given me the opportunity to pursue a PhD in the first place and has been most influential in shaping this thesis and myself as a scientist. His experience, guidance, and ideas throughout the last four years have made this research possible. I would also like to thank the DX community and other researchers I have met along the way for their input on my work.

As this thesis has emerged from the AMOR project, I would like to express my gratitude to everybody involved, especially Uptime Engineering; without their experience, insights, and commitment to the project, we would not have come this far. In addition, this thesis was partially funded by the Austrian Research Promotion Agency (FFG) under grant 842407.

My research would have been impossible without the aid and support of my students, who have either contributed directly to the content of this thesis or indirectly by supporting me in my teaching duties. Further, I cannot go without mentioning my colleagues at the Institute for Software Technology and other institutes/competence centers. Not only do I appreciate the scientific discussions and counseling, but first and foremost the non-work related conversations that put things in perspective and helped me through the ups and downs that is a PhD.

Heartfelt thanks go to my parents and brother, who have always motivated me to challenge myself and supported me during my entire life and studies.

More than anyone else I would like to thank my husband. This thesis would not have been possible without him. His encouragement, reassurance, and especially patience were indispensable in order to achieve this PhD thesis.

Contents

List of Figures

List of Tables

Abbreviations

ACD the Anticoincidence Detector mounted on the Large Area Telescope of the Fermi Gamma-ray Space Telescope.

AI Artificial Intelligence.

ALP Abductive Logic Programming.

AMOR Applied Model-Based Reasoning.

ASP Answer Set Programming.

ATMS Assumption-based Truth Maintenance System.

BBN Bayesian Belief Network.

BDD Binary Decision Diagram.

BHS-Tree Binary Hitting Set Tree.

CBR Case-based Reasoning.

CNF Conjunctive Normal Form.

DAG Directed Acyclic Graph.

DLP Disjunctive Logic Programming.

DNF Disjunctive Normal Form.

FDI Fault Detection and Isolation.

FMA Failure Mode Assessment.

FMEA Failure Mode Effect Analysis.

FMECA Failure Mode, Effects, and Criticality Analysis.

FMMEA Failure Modes, Mechanisms and Effects Analysis.

FTA Fault Tree Analysis.

GDE General Diagnosis Engine.

GUI Graphical User Interface.

HIFI-FPU Focal Plane Unit of the Heterodyne Instrument for the Far Infrared built for the Herschel Space Observatory.

HS-DAG Hitting Set Directed Acyclic Graph.

HST Hitting Set Tree.

IGBT insulated-gate bipolar transistor.

ILP Integer Linear Programming.

INDIA Intelligent Diagnosis in Industrial Applications.

LTUR Linear Time Horn Clause Theorem Prover.

MAP Maximum a posteriori assignment problem.

MCS Minimal Correction Subset.

MiTS the Maritim ITStandard.

MONET Model-based systems and qualitative reasoning.

MPE Most probable explanation.

MSS Maximal Satisfiable Subset.

MUS Minimal Unsatisfiable Subset.

O&M Operation and Maintenance.

OSFDP One Single Fault Diagnosis Property.

PAP Propositional Abduction Problem.

PCB printed circuit boards.

PoF Physics of Failure.

RPN Risk Priority Number.

SCADA Supervisory Control And Data Acquisition.

SFK-resolution Skip-Filtered Kernel-Resolution.

SOL-deduction Skipping Ordered Linear Deduction.

SOL-resolution Skipping Ordered Linear Resolution.

SOL-tableau calculus Skipping Ordered Linear tableau calculus.

WEKA Waikato Environment for Knowledge Analysis.

ZBDD Zero Suppressed Binary Decision Diagram.

Part I

The Basics

1 Introduction

> *Destiny guides our fortunes more favorably than we could have expected. Look there, Sancho Panza, my friend, and see those thirty or so wild giants, with whom I intend to do battle and kill each and all of them, so with their stolen booty we can begin to enrich ourselves. This is nobel, righteous warfare, for it is wonderfully useful to God to have such an evil race wiped from the face of the earth.*
>
> — **Miguel de Cervantes Saavedra**
> "*Don Quixote*". 1605.

1.1 Motivation

Accurate failure diagnosis in technical systems is a topic of interest from an industrial as well as research point of view and has grown in importance due to the increasing complexity and magnitude of such systems. On December 18, 2017, right around the corner of the winter holidays, Hartsfield-Jackson Atlanta International Airport faced an eleven-hour power outage causing Delta Air Lines to cancel around 1,400 flights. The utility provider identified the root of the failure as a failing piece of gear that caused a fire which also damaged the backup system. This incident may cost Delta Air Lines up to $50 million [Reu17]. Right around that time, the cost of Brent crude oil hit its highest level since 2015 as the news about a system breakdown of the Forties pipeline became public. The Forties pipeline is the only connection transporting oil extracted in the North Sea to the United Kingdom. Given the collapse of the supply channel originated from a hairline crack [Gua17] and the cold winter months, experts expected that the unscheduled maintenance of two to three weeks could seriously affect the country's energy supply [Gur17]. In the presence of a failure, diagnosing the origin of a malfunction is a pivotal precondition for any repair or replacement activity aiming at reestablishing the system's intended operation.

Diagnosis is a hard task requiring a considerable amount of expertise of subsystems, their interactions and behavior, as well as expertise of previous fault situations [CT06]. Hence, in the 1970s, the Artificial Intelligence (AI) community began to develop knowledge-based systems that assisted in the task of fault identification. These systems were based on expert knowledge of symptoms and their corresponding causes and have proven especially useful in the medical domain, such as MYCIN [BS84], which provides decision support for antibiotic therapies. Yet in large applications the interdependences between the rules defining the information led to considerable maintenance issues [Got+90]. To avoid these complications, model-based approaches have emerged in the nineteen-eighties. Model-based diagnosis exploits an abstract description of the underlying system in order to single out the cause for an observed anomaly. During the last decades a solid theoretical background has been developed with two approaches emerging: consistency-based and abductive diagnosis. Consistency-based diagnosis relies on a formalization of the correct system response and identifies anomalies through inconsistency [DKW87; Rei87]. The abductive version is founded in logic-based abduction, where the observation is a logical consequence of the explanation given the background knowledge of the system [CT06]. Thus, this type of

inference is especially suited for fault identification as it coincides with human diagnostic solving. An abductive diagnosis model considers information on failures and how they affect detectable system measurements. For technical systems this entails that the model depends on a systematic and analytic knowledge of components, their possible malfunctions, and the subsequent effects.

Model-based diagnosis has been applied to various domains, such as space probes [WN96] or software debugging [Fri+99]. Although it is an active research topic, where the field continuously improves the techniques and contributes with its results to other application areas as well, the use of this technology in industrial applications is still sparse. In recent years, there have been several attempts to carry model-based diagnosis techniques over to industrial applications. For instance, the Intelligent Diagnosis in Industrial Applications (INDIA) project aimed at integrating model-based diagnosis in industrial applications by analyzing how the theoretically developed techniques can be related to already accepted tools and systems [Mil+00]. The goals of the INDIA project were the transfer of theoretical results to real industrial domains, to direct research towards the issues uncovered, and finally drive science to create approaches that are applicable in practice. Within the project a model-based engine for real-time on-board automotive diagnosis was developed [Sac+00] as well as a tool for facilitating the construction of Failure Mode Effect Analyses (FMEAs) in the aeronautics domain [CR01]. Another approach to bridge the gap between science and practice has been the integrated diagnosis toolkit DiKe that encompasses different diagnosis engines, a Graphical User Interface (GUI) to promote user acceptance, and a language for formalizing diagnostic models that does not require a comprehensive education in logic [Fle+01]. On the European side, the Model-based systems and qualitative reasoning (MONET) initiative dedicated to the promotion of model-based techniques revealed certain barriers between industry and research [TMM98]:

- *Technical Gaps* are first and foremost the lack of automatic modeling tools and the difficulties of consolidating model-based tools and existing software.
- *Human Gaps* result from management or organization issues. These barriers include obstacles that concern technology acceptance and user interaction, the difference in expertise regarding modeling between human operators and the knowledge engineers, and the unavailability of tools allowing novices to rapidly build models to experiment with the techniques.
- *Economic Gaps* are, for instance, the absence of data on the financial benefits of implementing model-based tools and the need for approaches dealing with model evolution.

This thesis aims at providing notions about how to reduce these barriers. Our main goal is to develop a methodology for applying abductive model-based diagnosis to industrial practice. Particularly, we define a framework to integrate the industry-relevant task of diagnostics in the overall operational life of systems that allows for easy integration with existing tools. We show a general technique for consolidating diagnosis, explanation ranking, and measurement selection and provide a method for reducing/eliminating the initial modeling effort associated with model-based diagnosis. Moreover, in this thesis we investigate computational approaches to diagnostic problem-solving which allow us to derive explanations in reasonable time for real-world applications.

We work towards these objectives from two perspectives. On the one hand, we theoretically define a blueprint for intercalating abductive model-based diagnosis into practical operations. This framework explores the possibilities to rely on existing expert knowledge for compiling a system description suitable for fault identification and performs refinements to the initial diagnosis result. While the formalizations used as a diagnostic model can be quite diverse,

the models in our case comprise subsets of propositional logic. Based on the underlying system description, we reveal which abduction inference mechanism is advantageous.

As with any thesis that is located at the intersection of theory and practice, we seek to verify on a case study if the theory fits reality. To provide evidence that the methodology can be effectively used in industrial applications, we showcase the incorporation of an abductive modeling and diagnosis engine into the industrial wind turbine domain. The wind energy sector has been growing steadily as a source for electrical power generation around the world. With this expansion, the complexity of industrial wind turbine installations has increased significantly over the last decades [Ass14]. Service costs are high and turbine downtimes are associated with substantial revenue loss. Thus, in order to make wind a competitive force on the energy market it is critical to improve turbine reliability and optimize the maintenance process. As the currently implemented standard alarm systems deliver a large number of false positives, they are not well suited as a standalone fault identification tool [GW10]. Besides the economic considerations, wind turbine accidents have increased in recent years. In catastrophic situations an instantaneous bearing failure may cause an entire turbine blade to become separated from the hub, possibly leading to a blade beeing thrown into the surrounding areas [Rob+13]. Therefore in the second half of the thesis, we attend to the integration of an abductive diagnosis application into the wind turbine domain to optimize the maintenance and fault identification task. This research has been conducted in cooperation with our industrial partner on the Applied Model-Based Reasoning (AMOR) project.

1.2 Applied Model-Based Reasoning (AMOR) Project

The majority of the work presented in this thesis was conducted in the context of the AMOR project[1] (Austrian Funding Agency FFG under contract number 842407). AMOR aims at developing a methodology and framework for exploiting model-based diagnosis in daily industrial practice as well as produce a proof-of-concept diagnosis system to be of practical use.

To provide evidence of the feasibility and utility of the approach in a practical setting, the project utilizes the wind turbine industry as an application domain. Our industrial partner, Uptime Engineering[2], provides expertise as reliability experts and developers of condition monitoring software for wind power plants and conducts the evaluation of the project results within the given context.

1.3 Research Objectives and Contribution

This work is strongly influenced by the AMOR project; hence, all theoretical work is tightly connected to the goal of implementing abductive model-based diagnosis in real-world applications. Therefore, the main research objective can be phrased as follows:

RO *Develop a framework for facilitating the application of abductive model-based diagnosis in industrial practice.*

Adopting a divide and conquer strategy we break this research objective down into two parts: first, we examine the issue from a theoretical perspective and second, we take the findings

[1] http://www.ist.tugraz.at/amor/
[2] http://www.uptime-engineering.com/

from theory and consider them in an industrial context. Figure 1.1 depicts the research objectives as well as the chapters of this thesis concerned with achieving each goal.

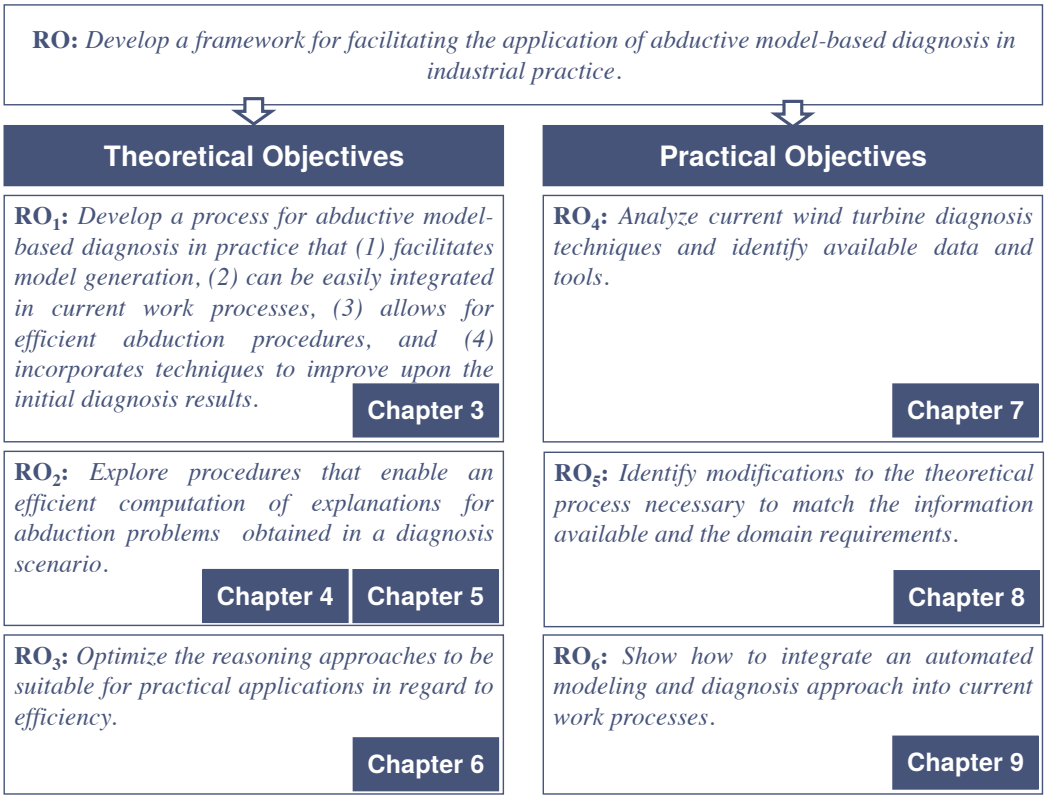


Figure 1.1: Research objectives addressed throughout the thesis.

Theoretical Objectives

RO$_1$ *Develop a process for abductive model-based diagnosis in practice that (1) facilitates model generation, (2) can be easily integrated into current work processes, (3) allows for efficient abduction procedures, and (4) incorporates techniques to improve upon the initial diagnosis results.*

As a starting point for the theoretical portion of this thesis, we have to develop a hypothesis about how such a methodology for incorporating model-based diagnosis in practice may present itself. In addition, it is essential that the gaps between research and real-world applications be contemplated and circumvented if possible. Thus, the goal of Chapter 3 is to establish a strategy suitable for industrial applications that incorporates solutions to the modeling dilemma, permits efficient diagnostic reasoning, and aids in limiting the number of diagnoses to process by the human operator.

RO$_2$ *Explore procedures that enable an efficient computation of explanations for abduction problems obtained in a diagnosis scenario.*

The fault identification task is, in general, a hard problem. To ensure user acceptance, however, diagnoses are to be computed in an acceptable time span. While acceptable in this context is application dependent, there is a strong need to determine a notion of the magnitude at which various abduction procedures derive explanations to ensure that given an explicit domain an approach with suitable runtime behavior can be selected. In Chapter 4 and 5 we study algorithms exploitable for the types of diagnosis problems we are concerned with to determine superior tools and tactics.

RO$_3$ *Optimize the reasoning approaches to be suitable for practical applications in regard to efficiency.*

To ensure that abduction is appropriate even in time-critical applications, we are interested in whether there are strategies we can adopt to improve the efficiency of the abduction approaches investigated previously. Chapter 6 achieves this objective by exploiting the concept of algorithm selection.

Practical Objectives

For the second portion, we rely on our industrial partner of the AMOR project and their domain expertise. In particular, we strive after providing a case study presenting our theoretically defined process in the industrial wind turbine domain. Again in order to achieve this goal, we define objectives at a deeper level of granularity.

RO$_4$ *Analyze current wind turbine diagnosis techniques and identify available data and tools.*

Our integration approach must consolidate model-based fault identification with existing software and facilitate the development of a diagnosis model. Initially, the available knowledge and tools are to be investigated in order to determine to what extent the previously defined methodology can be mapped to practice and what kind of modifications may be necessary. In Chapter 7, we give an overview of fault identification in the wind turbine domain and discuss the type of information at our disposal.

RO$_5$ *Identify modifications to the theoretical process necessary to match the information available and the domain requirements.*

Given the analysis of the initial situation, it is necessary to determine how the approach may be modified in order to fit the specifics of the application domain. Throughout the AMOR project, the underlying expressiveness of the failure assessment utilized for compiling the system descriptions changed. Thus, it is essential to handle these circumstances by adapting—in this case—the compilation approach in Chapter 8.

RO$_6$ *Show how to integrate an automated modeling and diagnosis approach into current work processes.*

To complete the case study on wind turbine diagnosis, we seek to show the manner in which we integrate the automated modeling approach and the abductive diagnosis engine into current software systems. Besides the technical solution, it is essential to consider factors that have been found to affect the acceptance of a software product. Thus, in Chapter 9 we provide a workflow that respects the current fault identification process of the service technicians maintaining the turbine and advocates for the importance of usability and usefulness.

By addressing the research objectives, we contribute to the fields of abductive reasoning, model-based diagnosis, and the application of knowledge-based systems in industry. More specifically, our main contributions are the following:

- We provide a methodology for abductive model-based diagnosis in an industrial setting. Suitable models are created automatically from failure assessments available and methods for improving upon the initial diagnosis results are given.
- By showing the equivalence between models generated on top of structured failure analyses and bipartite diagnosis problems in simple set-covering theory, we can utilize hitting set enumeration for abduction over this class of models. Through empirical evaluations, we show that for these types of models a hitting set approach is advantageous over other abductive reasoning methods.
- Furthermore, we analyze direct reasoning and conflict-driven techniques, which can be employed for abduction on propositional Horn clause models, theoretically and empirically.

- To improve the efficiency of abduction, we are the first to apply algorithm selection to model-based diagnosis. We present a meta-approach based on structural features of the diagnosis problems to choose the fastest technique for a new abduction instance.
- Given the developed methodology, we apply it to the industrial wind turbine domain, where we (1) provide adaptations to the diagnosis model in order to retrieve "intuitive" explanations more similar to human diagnostic reasoning, and (2) describe the realization of a model-based diagnosis tool under specific consideration of the current work processes as well as user interface and interaction design.

1.4 Outline

This thesis is organized as follows. In Part I we attend to the basics for this thesis, i.e., motivational aspects in Chapter 1 and preliminaries for the remaining work in Chapter 2. Part II is concerned with barriers to applying model-based diagnosis in practice: the lack of convenient modeling capabilities, the integration into current work processes, and the efficient computation of diagnoses. In particular, in Chapter 3 we define a process for incorporating abductive model-based diagnosis in practice that is based on the automatic compilation of failure assessments to system descriptions appropriate for fault identification. Chapter 4 and 5 examine abduction and suitable reasoning procedures within a subset of propositional logic, i.e., simple bipartite diagnosis problems and Horn clause abduction, respectively. Both chapters utilize different formulations of abductive reasoning and provide empirical comparisons of the presented algorithms. Based on these results, we investigate in Chapter 6 the possibility of improving abductive reasoning in terms of efficiency by exploiting the notion of algorithm selection. Part III covers the case study, in which we seek to apply our process and theoretical results to the industrial wind turbine domain. We open in Chapter 7 by giving some contextual information on wind turbine diagnosis, before discussing in Chapter 8 adaptations we have to perform to the model in order to provide intuitive explanations that are more on par with human-like diagnostic reasoning. Chapter 9 closes this part of the thesis and is concerned with applying the process defined in Chapter 3 to the wind turbine domain. We discuss the integration of an abductive modeling and diagnosis engine with special regard to the application's workflow, the user experience, and the GUI design to foster acceptance by the human operators. Lastly, in Part IV, we summarize the thesis, discuss limitations, and express open challenges that present themselves as possible future work.

Preliminaries

2

> *... abduction starts from the rule and guesses that the fact is a case under that rule: All people with tuberculosis have bumps; Mr. Jones has bumps; perhaps Mr. Jones has tuberculosis.*
>
> **— Warren S. McCulloch**
> *"What's in the brain that ink may character?"*. 1964.

This chapter is based on the following publications:

- [KW16a] Roxane Koitz and Franz Wotawa. „Improving Abductive Diagnosis Through Structural Features: A Meta-Approach". In: *Proceedings of the 2016 International Workshop on Defeasible and Ampliative Reasoning*. 2016, pp. 1–9

Parts of the definitions in Section 2.2.2 on Abduction by Set-covering have been published in [KW16a].

2.1 Diagnosis of Complex Systems

Fault[1] diagnosis is the process of identifying root causes of an incorrect system behavior based on observable symptoms. That is, a diagnostic system maps anomalies to defects. The tasks associated with diagnostic reasoning is are debatable; most work associates diagnosis with fault detection, fault identification, and/or repair [WS01]. In this thesis, the diagnosis is mainly concerned with fault identification. Determining the origin of a failure is a crucial prerequisite before any corrective action may be taken in order to restore a healthy system status. Automating this process has proven necessary especially in domains, (1) where the scale of systems and magnitude of fault sources complicates diagnosis, e.g., communication networks, (2) where systems are inherently complicated, e.g., spacecrafts, or (3) in cases where the presence of a failure is hazardous or linked to excessive revenue loss, e.g., industrial wind turbines [Sol+17].

Many industrial systems have monitoring software installed that allow the detection of anomalies; yet, these tools usually do not provide information on the failure location. This of course does not hold for systems where there is a one to one correspondence between failures and their manifestations. However, more commonly a symptom can be triggered by a diverse set of malfunctions and a single defect may affect several components within the system. The fundamental difficulty of diagnosis, is the complicated relation between failures, intermediate consequences, and observable effects. Unfortunately, automatic monitoring and troubleshooting systems often fail to correctly identify the occurrence of a failure or point towards a unit not responsible for the detected anomaly. Thus, diagnosis itself has been mostly a manual endeavor [Kav+12].

Without the claim of providing an exhaustive review on troubleshooting approaches, we want to give a notion of some common fault identification techniques. As stated by Dressler and Puppe [DP99] in knowledge-based fault identification there are three major knowledge types: (1) experience, (2) models, and (3) cases. Heuristic diagnosis systems are based

[1] While there is a difference between failure and fault for instance in the field of Root Cause Analysis, we use these terms synonymously throughout this thesis [Sol+17].

on experience of solving diagnosis problem. One of the commercially most successful approaches to AI are expert systems. Empirical expert knowledge is encoded as rules describing the relation between symptoms and failures. These instructions consist of a premise and a consequence, e.g., "`IF EngineDoesNotStart THEN PossibleCauseBatteryFlat`" [Str08]. Rule-based techniques provide a clear separation between domain knowledge and inference mechanisms and their inference is similar to human diagnostic reasoning. The diagnostic applications of expert systems range from the medical domain [BS84] over telecommunications networks [Bru+93] to wind turbine gearbox fault identification [ZL+12]. A main disadvantage is the structure of the knowledge representation that renders these applications hard to maintain; in large systems rules are usually interdependent and hence, modifying a single instruction may cause unexpected effects. Additionally, given a new fault to consider within the diagnostic application, all rules with a relevant symptom as premise have to be adapted [Got+90]. Another issue, as pointed out by Struss [Str08], is that the nature of the knowledge is empirical and thus depends on the context it was produced in. That means that the model may implicitly contain information on structure, which may change given different compositions of the system components, and that the model is created solely for the purpose of diagnosis making reuse for other problems difficult to even impossible.

Once the disadvantages of the rule-based approach were discovered, model-based diagnosis followed as a way to assist humans in fault-identification and decision-making tasks. Model-based diagnosis' main idea is to reason from "first principles", i.e., the representation of the system encompasses information about its structure (static knowledge) and functional behavioral (dynamic knowledge). As with rule-based systems, there is a separation between the domain and task knowledge, however, an essential benefit of the model-based approach should be the possibility to use a component library. This library contains models of different units that can be connected together into various comprehensive system descriptions. Different representation formalisms have been proposed such as logic, constraints, quantitative mathematical equations, or qualitative models [Str08]. There are two variations of the model-based diagnosis: given the system description and a set of symptoms, we can derive the root causes either through the observations contradicting the model (consistency-based diagnosis) or the observations following from the model (abductive model-based diagnosis) [Str08]. Importantly, model-based diagnosis relies on a comprehensive collection of faults on an appropriate abstraction level [DP99]. Besides AI theFault Detection and Isolation (FDI) community of Control Engineering has developed model-based approaches. Based on a system model formalizing the behavior of components, it is checked whether the sensor observations are consistent with the model, i.e., the residual is derived. The residual is the difference between the measured process variables and their estimates. Cordier et al. [Cor+04] have shown the links between the FDI and the consistency-based approach by investigating their relations and developing a unified framework. In Section 2.3, we provide more information on model-based diagnosis with a special focus on the abductive variant.

In the absence of a model and expert knowledge, there may be a collection of reports on previous failure incidents and their maintenance that can be utilized for Case-based Reasoning (CBR). Case-based reasoners solve new problems by using or adapting solutions that were employed to answer prior queries. As with the rule-based strategy this type of fault diagnosis offers a reasoning paradigm that is similar to the way humans routinely solve problems. Althoff et al. [Alt+89] present the early CBR system PATDEX used in the troubleshooting of Computerized Numeric Control machining centers. The PATDEX architecture is based on diagnostic cases. Solutions are derived based on a similarity measure and the application takes into account symptom relevance and test selection. Portinale, Magro, and Torasso [Por+04] present a system combining case-based and model-based reasoning. Whenever a new diagnosis problem is applied, a suitable solution from the case

memory is fetched and adapted if necessary based on behavioral models of the system. Hence the case-based reasoning portion aims at speeding up the model-based diagnosis part. Other diagnosis approaches include statistical classification methods, such as belief networks that are suitable when dealing with incomplete information or lack of data, neural classifications that exploit knowledge implicitly provided by a large number of samples, or decision trees that encode actions, decisions and their consequences [DP99].

2.2 Abductive Reasoning

Abduction reasoning originates from philosophy where it has been introduced by Peirce [Pei74] as the only synthetic reasoning method, i.e., the sole procedure capable of suppling new ideas. This type of inference derives a minor premise (*case*) given the major premise (*rule*) and conclusion (*result*) [Ino02].

Example 2.1 **[Ino02]**

(*rule*)	All the beans from this bag are white.
(*result*)	These beans are white.
(*case*)	These beans are from this bag.

It was later rediscovered for AI as another inference type since deduction has been found inadequate for many problem solving activities [Pop73]. In essence, abductive reasoning aims at generating a set of plausible explanations for given observations. We can describe abduction by the following inference rule [Pau93]:

$$\frac{\psi \rightarrow \phi, \phi}{\psi}$$

That is, given a fact ϕ in conjunction with a rule $\psi \rightarrow \phi$, an explanation ψ is hypothesized accounting for ϕ. Yet abduction does not characterize the relation between ψ and ϕ, i.e., causality as suggested by the material implication is not a necessary condition.

Abduction is an "ampliative" reasoning strategy that allows to draw conclusions beyond what is already contained in the premises, i.e., the conclusions derived are not necessarily true, but rationally justified. Hence, abductive reasoning is a "probable" inference [Ino02]. Given these characteristics, abduction represents a non-monotoning reasoning type, i.e., an inferred explanation may become invalid given additional observations [EG95]. The process of abduction is connected on the one hand to hypothesis generation and on the other hand to a selection of admissible explanations. Typically, a consistency requirement is enforced upon the explanations in addition to quality measure classifying a "best" solution. The notion of "best" is related to a preference criteria, such as Occam's razor, which accepts the simplest explanations [PR90].

Pople's [Pop73] work is one of the first on abduction in the field of AI providing a formulation of abductive reasoning in a theorem proving framework. Many publications followed afterwards; one of the first abduction systems was the medical diagnosis application INTERNIST-I performing medical diagnosis based on causal links between diseases and symptoms [Mil+82]. Cox and Pietrzykowski [CP86] present an algorithm for computing minimal and nontrivial causes for events given a knowledge base. Their approach is similar to Pople's [Pop73] by deriving explanations from the dead ends of failed resolution proofs on the background theory and negated symptoms. Similarly, the framework Theorist [Poo+87] augments the background theory with the observations and abducible primitives to derive

contradictions. Theorist [Poo+87] is a logic programming system based on clausal form of first order logic that uses abduction to perform several tasks such as default reasoning or diagnosis. Later, Levesque [Lev89] presents a different variant of abduction based on belief. His knowledge-level account is independent of the chosen knowledge representation, but characterizes it on the symbol level as a formal model of belief. Eshghi and Kowalski [EK89] relate abduction to negation as failure in the context of logic programming. The authors translate a logic program into an abduction framework. Later based on this work, abductive logic programming emerged [Kak+92]. In abductive logic programs, additional integrity constraints and abducible predicates are used to define the fragments and their conditions allowed to constitute solutions. While diagnosis is the most prominent application field for abductive reasoning, its operation has been utilized for several applications such as natural language understanding [Hob+93], test case generation [McI94], or ontology debugging [WK+14]. The body of research on abductive reasoning is extensive. Hence, we focus in the subsequent sections on different types of abduction, and discuss related literature in more detail there[2]. For overviews and reviews on abduction we refer to Paul [Pau93], Eiter and Gottlob [EG95], and McIlraith [McI98], who provide excellent work on abductive reasoning in AI.

Given background knowledge about a domain of discourse, an abduction problem is characterized as the search for a set of elements taken from a sub-vocabulary of the representation language that explains a given set of observed facts. This sub-vocabulary specifies the fragments of the language that can be considered within explanations, i.e., are abducible[3]. These vocables are often referred to as assumptions or hypotheses, which should stress the fact that this type of inference generates "probable" solutions rather than definite explanations. We take the formal definition of an abduction problem from Eiter and Gottlob [EG95]:

Definition 2.1 (Abduction Problem [EG95]). An abduction problem is a 4-tuple ($\mathcal{T}$, $\mathcal{H}$, $\mathcal{M}$[4], $\mathcal{M}^*$), where

- $\mathcal{T}$ is the background theory representing the domain knowledge,
- $\mathcal{H}$ denotes the set of abducible primitives (i.e., assumptions or hypotheses),
- $\mathcal{M}$ is a set of manifestations (also referred to as effects or symptoms), and
- $\mathcal{M}^* \subseteq \mathcal{M}$ describes the set of observed manifestations.

There are many formalizations of abduction within AI. Similar to previous reviews, we distinguish three main characterizations [Pau93; EG95; McI98]:

- Logic-based abduction
- Abduction by set-covering
- Probabilistic abduction

In our analysis, we focus mainly on logic-based and set-covering abduction, and only discuss the basics of the probabilistic perspective[5]. The remaining Chapter is structured as follows: First, we discuss logic-based abduction and relate it to prime implicates/prime implicants, consequence finding, and proof-tree completion. Then in Section 2.2.2, we examine abduction by set-covering before we give a short overview of probabilistic abduction.

[2] Note that our literature review is not comprehensive, but discusses some of the most influential works.
[3] Determining which parts of the language should be abducible is a problem itself, since an abducible may be appropriate in one scenario but not in another.
[4] Often $\mathcal{H}$ and $\mathcal{M}$ are defined as disjoint sets.
[5] Of course there are intersections between these formalizations and probabilistic reasoning has been considered within logic and set-covering abduction.

2.2.1 Logic-based Abduction

An extensive amount of research has concerned itself with abduction based on a logical model, which is an especially suitable problem formulation if the domain knowledge can be easily described by logical sentences. This is, for instance, the case when the relation between causes and their effects is available. However, logics can further represent knowledge containing disjunctive information or integrity constraints [EG95].

In logic-based abduction, the background knowledge $\mathcal{T}$ consists of logical sentences, describing the relations between variables. Hypotheses are a subset of the variables that are allowed to constitute solutions to the abduction problem. In this thesis, we focus on the propositional case and assume standard definitions for propositional logic [Ino91; Bit08]. A propositional formula ϕ in Conjunctive Normal Form (CNF), defined over a set of Boolean variables (or atoms) $X = \{x_1, x_2, \ldots x_n\}$, is a conjunction of m clauses $(C_1, C_2, \ldots, C_m)$. The Boolean constants *true* ($\top$) and *false* ($\bot$) as well as the connectives $\neg, \wedge, \vee, \rightarrow$ are interpreted in the standard way. A clause $C_i = (l_1, l_2, \ldots, l_k)$ is a disjunction of literals, where each literal l is either a Boolean variable or its complement. $L^{\pm}$ denotes the set of literals of ϕ, while L^{+} (L^{-}) represents the set of positive (negative) literals. A conjunctive clause or term $C_i' = (l_1, l_2, \ldots, l_k)$ is a conjunction of literals, where each literal l is either a Boolean variable or its complement. A truth assignment is a mapping $\mu : X \Rightarrow \{0, 1\}$ and a satisfying assignment for ϕ is a truth assignment μ such that ϕ evaluates to 1 under μ. Hence, a Propositional Abduction Problem (PAP) is defined as:

Definition 2.2 (Propositional Abduction Problem (PAP) [EG95]). A tuple $\langle \mathit{PROPS}, \mathcal{H}, \mathcal{M}^*, \mathcal{T} \rangle$ forms a PAP, where *PROPS* is a finite set of propositional variables, $\mathcal{H} \subseteq \mathit{PROPS}$ is the set of hypotheses, $\mathcal{M}^* \subseteq \mathit{PROPS}$ is the set of observed manifestations, and $\mathcal{T}$ is a propositional formula of the background knowledge, i.e., the theory.

Logic-based abduction provides an intuitive notion of explanation, i.e., a set of hypotheses is an explanation for an observed symptom if the observation is a logical consequence of the explanation given the background knowledge. A conclusion ϑ is said to be a logical consequence of a set of premises ψ, if and only if for any interpretation in which ψ is true ϑ is also true. We write this relation as $\psi \models \vartheta$ and say ψ entails ϑ.

Definition 2.3 (Solution; Abductive Explanation [EG95]). A solution or abductive explanation to PAP is a set Σ such that

(1) $\Sigma \subseteq \mathcal{H}$,

(2) $\mathcal{T} \cup \Sigma \models \mathcal{M}^*$, and

(3) $\mathcal{T} \cup \Sigma \not\models \bot$.

Sol(PAP) is the set containing all solutions to the PAP.

Abduction within the logic-based framework is defined via consistency and deriveability; the observed manifestation must be derivable from $\mathcal{T}$ augmented with the explanation Σ (condition **(2)**), while $\mathcal{T} \cup \Sigma$ is satisfiable (condition **(3)**). In addition, an explanation can only contain elements from the set of hypotheses (condition **(1)**). Since the goal of abduction is to derive the "best" explanations, some preference criteria must be present to characterize the notion of optimality. A common limitation is to only consider subset-minimal explanations[6].

[6] At least in the context of diagnosis, this is a preferred feature. However, other application domains may call for different optimality criteria. For instance, in learning applications a more general solution may be preferred [McI98].

Definition 2.4 (Minimal Abductive Explanation). An explanation Σ is minimal iff there is no other abductive explanation Σ' such that $\Sigma' \subseteq \Sigma$.

Example 2.2

Consider the following knowledge (inspired by Pearl [Pea88]):

- We know that when the sprinkler is on, that the grass is wet.
- When the sprinkler is off, it is quiet.
- The sprinkler cannot be on and off at the same time.
- When it rains, not only the grass is wet, but also the street.
- Wet grass and sunshine leads to grass growth.

Given this information, we can construct a simple logical theory

$$\mathcal{T} = \left\{ \begin{array}{c} sprinkler\_on \rightarrow wet\_grass, sprinkler\_off \rightarrow quiet, \\ \neg sprinkler\_on \vee \neg sprinkler\_off, rain \rightarrow wet\_grass \wedge wet\_street, \\ wet\_grass \wedge sunshine \rightarrow growing\_grass \end{array} \right\} .$$

The proposition variables are $PROPS = \{sprinkler\_on, wet\_grass, sprinkler\_off, quiet, rain, wet\_street, sunshine, growing\_grass\}$. In addition, we define which variables are abducible, i.e., $\mathcal{H} = \{sprinkler\_on, sprinkler\_off, rain, sunshine\}$.

Assume that we see the grass is wet and it is quite, i.e., $\mathcal{M}^* = \{wet\_grass, quiet\}$. To compute the explanations for our phenomena, we have to find sets of elements of $\mathcal{H}$ which together with $\mathcal{T}$ entail the observations and are consistent with the theory. Considering the entailment relations between assumptions and the observations, we can see that

$$\mathcal{T} \cup \{sprinkler\_off, rain\} \models \{wet\_grass, quiet\}$$

and

$$\mathcal{T} \cup \{sprinkler\_off, sprinkler\_on\} \models \{wet\_grass, quiet\} .$$

However, the second explanation candidate $\{sprinkler\_off, sprinkler\_on\}$ leads to a contradiction given the theory, since the sprinkler cannot be on and off simultaneously, i.e., the solution is inconsistent and does not constitute an abductive explanation. Thus, the only minimal explanation for our observations is $\Sigma = \{sprinkler\_off, rain\}$.

Prime Implicates and Prime Implicants

The concepts of prime implicate and prime implicant take an important role within logic-based abduction and first have been considered in the context of circuit simplification in the 1950s [Qui55].

Definition 2.5 ((Prime) Implicate [Mar00]). Given a set of propositional formulas Ψ, a disjunction of literals π is an implicate of Ψ iff $\Psi \models \pi$ and π is not a tautology. π is a prime implicate of Ψ iff

- π is an implicate of Ψ, and
- for every implicate π' of Ψ, if $\pi' \models \pi$ then $\pi \models \pi'$.

As Definition 2.5 declares, an implicate π is a consequence of Ψ, with a prime implicate being the strongest logical consequence. Note that a tautology has no prime implicates, and

the empty clause is the only prime implicate of a contradiction [Mar00]. Similarly, the dual of prime implicates, the so called prime implicants are defined:

Definition 2.6 ((Prime) Implicant [Mar00]). Given a set of propositional formulas Ψ, a conjunction of literals ϖ is an implicant of Ψ iff $\varpi \models \Psi$ and ϖ is not a contradiction. ϖ is a prime implicant of Ψ iff

- ϖ is an implicant of Ψ, and
- for every implicant ϖ' of Ψ, if $\varpi \models \varpi'$ then $\varpi' \models \varpi$.

Proposition 1 (Duality of prime implicates and prime implicants [Mar00]). A clause π is a prime implicate of Ψ iff $\neg\pi$ is a prime implicant of $\neg\Psi$. A conjunctive clause ϖ is a prime implicant of Ψ iff $\neg\varpi$ is a prime implicate of $\neg\Psi$.

Example 2.3

Let $\Psi = (a \vee b \vee c) \wedge (\neg a \vee b)$. Ψ has two prime implicates, i.e., $\pi_{\Psi_1} = (\neg a \vee b)$ and $\pi_{\Psi_2} = (b \vee c)$, and two prime implicants, i.e., $\varpi_{\Psi_1} = (b)$ and $\varpi_{\Psi_2} = (\neg a \wedge c)$. Deriving the negation of Ψ we obtain $\neg\Psi = (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (\neg b) \wedge (a \vee \neg c) \wedge (\neg b \vee \neg c)$. Again $\neg\Psi$ has two prime implicates, i.e., $\pi_{\neg\Psi_1} = (\neg b)$ and $\pi_{\neg\Psi_2} = (a \vee \neg c)$, and two prime implicants, i.e., $\varpi_{\neg\Psi_1} = (a \wedge \neg b)$ and $\varpi_{\neg\Psi_2} = (\neg b \wedge \neg c)$. Considering the dual relation between prime implicates and implicants, we can determine the following equivalences:

$$\begin{array}{lcl} \pi_{\Psi_1} = (\neg a \vee b) & \equiv & \neg\varpi_{\neg\Psi_1} = (\neg a \vee b) \\ \pi_{\Psi_2} = (b \vee c) & \equiv & \neg\varpi_{\neg\Psi_2} = (b \vee c) \\ \varpi_{\Psi_1} = (b) & \equiv & \neg\pi_{\neg\Psi_1} = (b) \\ \varpi_{\Psi_2} = (\neg a \wedge c) & \equiv & \neg\pi_{\neg\Psi_2} = (\neg a \wedge c) \end{array}$$

Proposition 2 (Equivalence [Bie09]). Given a formula Ψ. The conjunction of the prime implicates and the disjunction of the prime implicants of Ψ are equivalent and both are equivalent to the formula itself.

Example 2.3 (cont.)

For $\Psi = (a \vee b \vee c) \wedge (\neg a \vee b)$ it is apparent that $(a \vee b \vee c) \wedge (\neg a \vee b) \equiv (b) \vee (\neg a \wedge c) \equiv (\neg a \vee b) \wedge (b \vee c)$.

Proposition 1 states prime implicates and the prime implicants are a representation of the original formula, i.e., there is not information loss in replacing a formula by its prime implicates and prime implicants [Bie09]. A minimal Disjunctive Normal Form (DNF) representation of a function Ψ is a disjunction of some of its prime implicants, while a minimal CNF Ψ is a conjunction of some of its prime implicates. While Proposition 1 enables us to utilize any algorithm for computing prime implicates in order to derive prime implicants and vice versa. Computing prime implicates is an NP-hard problem and has first been studied in detail in the middle of the previous century as a means to simplify truth functions and subsequently circuits. The first approaches to deriving prime implicates of a CNF formula are based on propositional resolution [Qui55; Tis67; KT90]; however, not all resolution strategies are suitable for deriving prime implicates. As stated by Marquis [Mar00] given $\Psi \models \gamma$ a resolution strategy $\mathcal{R}$ is

- proof-finding complete iff $\Psi \wedge \neg\gamma \vdash_R \square$.
- deduction (consequence-finding) complete iff $\exists\zeta : \Psi \vdash_R \zeta$ and $\zeta \models \gamma$, where ζ is a clause.

Given a deduction complete resolution strategy, we can derive all prime implicates: if π is an implicate of Ψ, there is a clause π^* that subsumes π and can be derived from Ψ by resolution. The prime implicates can be obtained by simplifying the set of clauses to keep only the logically strongest ones [Mar00].

Resolution-based methods construct the resolvent of two clauses and subsequently remove all clauses subsumed by the newly generated clause. Once all resolvents have been derived, i.e., there are no more changes, all prime implicates have been enumerated. First, Quine [Qui55] proposes an algorithm in the context of minimizing Boolean functions. His method iteratively identifies consensus; the consensus of two variables that can be resolved is their resolvent, unless the resolvent is a tautology, i.e., a consensus proof is resolution with tautology elimination [Kle92]. Algorithm BRUTE FORCE taken from de Kleer [Kle92] describes the brute force approach to the resolution-based technique. First, the result set Π is empty. Whenever, a new clause ψ of Ψ is removed from Ψ and considered for computation, it is checked for subsumption with the result set. If ψ is subsumed by any clause in Π then ψ is skipped (Step 3). Otherwise, all clauses in Π subsumed by ψ are removed from the result set. Then the algorithm tries to resolve ψ with every clause in Π and all derived resolvents are added to the set of clauses Ψ. Lastly, ψ is added to Π in Step 6 before continuing the execution with Step 2.

Algorithm 2.1: BRUTE FORCE [Kle92]

1 Let Ψ be a set of clauses for which the prime implicates are to be computed and let Π (the result) be $\emptyset$.

2 Take the first clause ψ from Ψ. If there are no more clauses in Ψ return Π.

3 If ψ is subsumed by any clause in Π, then go to Step 2.

4 Remove all clauses of Π which are subsumed by ψ.

5 Try to derive to resolve ψ with every clause in Π. Whenever there is a resolvent and it is not a tautology, add it to Ψ.

6 Add ψ to Π and go to Step 2.

Example 2.3 (cont.)

Consider again $\Psi = \{(a \vee b \vee c), (\neg a \vee b)\}$. Table 2.1 depicts the execution of BRUTE FORCE. The first clause considered is $(a \vee b \vee c)$. Since it is neither subsumed nor subsumes any clause in Π and cannot be resolved, it is simply added to Π. The second clause $(\neg a \vee b)$ can be resolved with a clause in Π, i.e., $(a \vee b \vee c)$. The resolvent $(b \vee c)$ is added to Ψ and $(\neg a \vee b)$ is added to Π. The next clause in Ψ is the resolvent $(b \vee c)$. Since $(b \vee c)$ subsumes $(a \vee b \vee c)$, $(a \vee b \vee c)$ is deleted from Π and $(b \vee c)$ is added to the result set. As Ψ contains no more elements, the final set of prime implicates is $\Pi = \{(\neg a \vee b), (b \vee c)\}$.

Table 2.1: Execution of BRUTE FORCE.

Ψ	Π	ψ	**action**
$\{(a \vee b \vee c), (\neg a \vee b)\}$	$\emptyset$	$(a \vee b \vee c)$	add $(a \vee b \vee c)$ to Π
$\{(\neg a \vee b)\}$	$\{(a \vee b \vee c))\}$	$(\neg a \vee b)$	resolve $(\neg a \vee b)$ with $(a \vee b \vee c)$ add $(b \vee c)$ to Ψ add $(\neg a \vee b)$ to Π
$\{(b \vee c)\}$	$\{(a \vee b \vee c), (\neg a \vee b)\}$	$(b \vee c)$	remove $(a \vee b \vee c)$ from Π add $(b \vee c)$ to Π

$\{\}$	$\{(\neg a \vee b), (b \vee c)\}$	return Π

The issue with this technique is that several unnecessary resolvents may be generated. Ordered resolution has been proposed as a means to reduce the number of redundant resolutions steps by defining a total ordering over the variables of the formula. Tison's algorithm [Tis67] and the improvements by Kean and Tsiknis [KT90] avoid many redundant actions by ordering the symbols, iterating over them, and performing all resolutions of the current symbol before moving on to the next variable. This ensures that possible resolutions that may appear later using said symbol result in redundant resolvents.

***Example 2.4* [Kle92]**

Let $\Psi = (a \vee b) \wedge (\neg b \vee c) \wedge (\neg c \vee d)$, and assume a variable ordering $a > b > c > d$. Figure 2.1 depicts the resolution steps. First, we try to resolve using a. Since there is no possible actions, we continue with variable b and use the first and second clause to derive $a \vee c$. For c there are two resolution steps, i.e., ② and ③ in Figure 2.1. Afterwards we could resolve $\neg b \vee d$ with the first clause using b, however, the variable b has already been processed. Hence, we can safely skip this resolution as this would result again in $a \vee d$ (as shown with dashed lines in Figure 2.1).

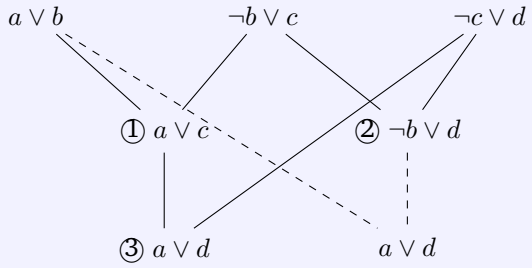


Figure 2.1: Tison's strategy to avoid redundant resolutions. [Kle92]

Based on Kean and Tsiknis' technique [KT90], the Complete Logic-based Truth Maintenance System [Kle92] is an incremental algorithm building discrimination trees (tries) in order to compute prime implicates with an improved subsumption check. A trie is a tree where edges are literals and the nodes along each branch represent the literals of a clause in descending order. The conjunction of all such clauses is a CNF equivalent to the formula. The algorithm uses the trie structure that allows on the one hand a more compact representation of the canonical form of clauses and on the other hand more efficient subsumption operations. Based on this work prime implicate tries were developed as an extension of tries, dedicated to store prime implicates [Mat+09].

Slage et al. [Sla+70] introduce an approach based on decomposition of the formula in DNF. Their tree method (Algorithm TREE METHOD) utilizes an ordered semantic tree, where nodes correspond to set of clauses and edges represent literals, to derive prime implicates. By computing success (✓) and failure leafs (✗), the semantic tree encodes implicates as paths from the root node to a successful leaf. Slage et al. [Sla+70] suggest the use of depth-first search to process the tree. The algorithm may possibly generate non-prime implicates, hence an additional subsumption elimination is necessary to compute only the prime implicates.

Algorithm 2.2: TREE METHOD [Sla+70]

1 Let Ψ_F be the set constructed from Ψ by removing tautologies. Let the root node be S and choose a literal ordering O_S (e.g., a frequency ordering of S).

2 If all leafs are marked either ✓ or ✗, return for each success node the path from the root to the node as a disjunction of the literals represented by the edges on the path. These disjunctions then correspond to the implicates. Otherwise, if there is a nonterminating node S_i, create for each literal l in O_S an edge from the node S_i to a node S_j. S_j holds all of S_i's clauses that do not contain l represented by the edge.
   - If S_j is empty, mark it with ✓.
   - Otherwise delete all $\neg l$ and all literals appearing in O_S before l. Afterwards, if S_j contains the empty clause, we mark S_j with ✗.

3 Repeat Step 3.

Example 2.3 (cont.)

Given $\Psi = \{(a \vee b \vee c) \wedge (\neg a \vee b)\}$ we construct the DNF representation, i.e., $\Psi_{DNF} = (\neg a \wedge c) \vee (b)$. We use a literal ordering O_S with $\neg a > c > b$. Starting from the root node S representing all clauses in Ψ_{DNF} for each literal in Ψ_{DNF} an edge is constructed. Node S_1 then contains all terms that do not contain the literal $\neg a$, i.e., (b). Processing S_1 all literals with a lower index than $\neg a$ in O_S are removed, which in this case are none. Thus the node contains (b) (see Table 2.2). Continuing in a depth-first order, we construct an edge from S_1 labeled b to node S_2. For S_2 there is no clause that does not contain the literal b characterizing the edge, hence, is empty and thus a success node and marked ✓. Hence, S_2 is a leaf and the algorithm continues with the next edge from the root labeled b. Node S_3 contains all terms that do not contain b, that is $(\neg a \wedge c)$. Since $\neg a$ and c have a higher index than b in O_S the literals are removed, S_3 contains the empty clause and the node is marked ✗ as shown in Figure 2.2. For the node S_4 and S_5 we continue analogously to nodes S_1 and S_2. Once all leafs are marked ✓ or ✗, we return for each ✓ leaf the path from the root to the node as a disjunction of the literals represented by the edges on the path. That is the prime implicates are $\neg a \vee b$ and $b \vee c$.

Table 2.2: Nodes content after the execution of TREE METHOD for Ψ_{DNF}.

S	$\{(\neg a \wedge c), (b)\}$	
S_1	$\{(b)\}$	
S_2	$\emptyset$	✓
S_3	$\{(\cancel{\neg a} \wedge c \cancel{)}\} = \{\}$	✗
S_4	$\{(b)\}$	
S_5	$\emptyset$	✓

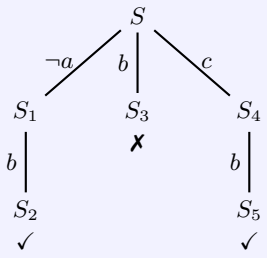


Figure 2.2: Tree method for Ψ_{DNF}.

Given the duality of prime implicates and prime implicants, we can utilize the tree method to derive prime implicants on the Ψ. We considering a frequency order O_S with $b > c > a > \neg a$. Starting from the root node S representing all clauses in Ψ for each literal in Ψ an edge is constructed. Node S_1 then contains all clauses that do not contain the literal a, i.e., $(\neg a \vee b)$. Processing S_1, first, $\neg a$ has to be removed since it is complimentary to a, and second, also all literal which have a lower index than a in O_S are removed, i.e.,

b is deleted from S_1 (see Table 2.3). As S_1 now contains the empty clause, the node is marked ✗ as shown in Figure 2.3. For S_2 there is no clause that does not contain the literal b characterizing the edge, hence, S_2 is empty and thus a success node and marked ✓. In node S_3 we remove b, since c has a higher index than b in O_S. Thus, the remaining clause is $\neg a$. Since the node is neither marked a success nor a failure, a new edge is constructed and labeled $\neg a$. The end point of the edge is node S_4, which is empty and hence marked as successful. Lastly, the node S_5 can be marked a failure analogous to S_1. Since all leafs are terminated, we can obtain the prime implicants from the paths of the success nodes as conjunctions of the literals, i.e., for S_2 we return b and for S_4 we return $c \wedge \neg a$.

Table 2.3: Nodes based on Ψ after the execution of TREE METHOD.

S	$\{(a \vee b \vee c), (\neg a \vee b)\}$	
S_1	$\{(\cancel{\neg a} \vee \cancel{b})\} = \{\}$	✗
S_2	$\emptyset$	✓
S_3	$\{(\neg a \vee \cancel{b})\} = \{(\neg a)\}$	
S_4	$\emptyset$	✓
S_5	$\{(\cancel{a} \vee \cancel{b} \vee \cancel{c})\} = \{\}$	✗

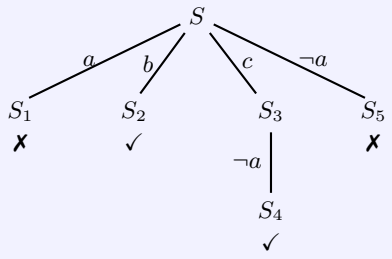


Figure 2.3: Tree method for Ψ.

Other approaches, are formulate to derive the prime implicants of a formula. For instance, Manquinho et al. [Man+97] present a method using an Integer Linear Programming (ILP) representation of a propositional formula in order to compute minimum-size prime implicants, being the prime implicant with the least number of literals. Bittencourt [Bit08] introduces quantum notation, a prime form notation that explicitly represents the relationship between literals, clauses, and terms. Via A^* search the state space is transversed and each state is associated with a set of quanta representing an incomplete prime implicant. Rymon [Rym94] has formalized the connection between prime implicants and minimal hitting sets as a means to generate prime implicants/implicates.

Definition 2.7 (Hitting Set [Pil+11]). Given a set of sets *CS*, a set $h \subseteq \bigcup_{C_i \in CS} C_i$ is a hitting set for *CS*, iff for any set $C_i \in CS : h \cap C_i \neq \emptyset$. A hitting set h is said to be minimal iff there exists no other hitting set h' for *CS* such that $h' \subset h$.

Proposition 3 (Prime Implicants/Implicates as Minimal Hitting Sets [Rym94]). Given a set of propositional formulas Ψ in CNF, a conjunction of literals ϖ is a non-trivial prime implicant of Ψ iff ϖ (as a set) is a minimal hitting set for Ψ (as a collection of sets) and it does not contain a literal and its negation. Similarly, given a set of propositional formulas Ψ_{DNF} in DNF a disjunction of literals π is a non-trivial prime implicate of Ψ_{DNF} iff π (as a set) is a minimal hitting set for Ψ_{DNF} (as a collection of sets) and it does not contain a literal and its negation.

Example 2.3 (cont.)

Consider again $\Psi = (a \vee b \vee c) \wedge (\neg a \vee b)$. Viewing Ψ as a collection of sets, we can define $CS = \{\{a, b, c\}, \{\neg a, b\}\}$. Computing the minimal hitting set for *CS*, we obtain $\{b\}$, $\{\neg a, c\}$, and $\{\neg a, a\}$. The last minimal hitting set, however, contains a literals and its complement, thus we do not consider it. Hence, $\{b\}$ and $\{\neg a, c\}$ are the set representations of the two prime implicants of Ψ.

For $\Psi_{DNF} = (\neg a \wedge c) \vee (b)$, we derive the minimal hitting sets for $CS = \{\{\neg a, c\}, \{b\}\}$, which are $\{\neg a, b\}$ and $\{b, c\}$ and hence the set representation of the prime implicates.

Now it remains to show how the notion of prime implicates/prime implicants be used for abduction. Considering Definition 2.3 of a solution to a PAP, we know that $\mathcal{T} \wedge \Sigma \models \mathcal{M}^*$ in order for Σ to constitute an abductive explanation. Rewriting this relation in an entailment preserving way, we obtain $\mathcal{T} \wedge \neg\mathcal{M}^* \models \neg\Sigma$, that is $\neg\Sigma$ is a prime implicate of $\mathcal{T} \wedge \neg\mathcal{M}^*$. Hence, we can establish the following proposition:

Proposition 4. Let $\mathcal{T}$ be the background knowledge and Ψ be a formula, then a clause Σ is a minimal abductive explanation of Ψ iff the clause $\neg\Sigma$ is a prime implicate of $\mathcal{T} \wedge \neg\mathcal{M}^*$[7] and satisfies conditions **(1)** and **(3)** of Definition 2.3.

Example 2.5

Consider the following PAP with $PROPS = \{sprinkler\_on, wet\_grass, rain, \text{wet\_street}\}$, $\mathcal{H} = \{sprinkler\_on, rain, \}$, $\mathcal{M}^* = \{wet\_grass\}$, $\mathcal{T} = \{\neg sprinkler\_on \vee wet\_grass, \neg rain \vee wet\_grass, \neg rain \vee wet\_street\}$.

We can create $\mathcal{T} \wedge \neg\mathcal{M}^* = \{\neg sprinkler\_on \vee wet\_grass, \neg rain \vee wet\_grass, \neg rain \vee wet\_street, \neg wet\_grass\}$. Given *wet_grass* is false, $\neg$*sprinkler_on* and $\neg$*rain* have to be true, i.e., are entailed by $\mathcal{T} \wedge \neg\mathcal{M}^*$. Hence, *sprinkler_on* and *rain* are explanations for the PAP.

Besides the abductive context, the importance of prime implicates in the context of consistency-based diagnosis has been established. There conflicts are the prime implicates of the system description and observations, while their hitting set dual prime implicants determine the diagnoses [DKW87].

Consequence Finding

Consequence finding represents a general reasoning scheme for obtaining solutions to a variety of problems, such as prime implicate computation [Mar00]. In essence, the corresponding methods infer logical consequences from a given knowledge base, while restricting the generated solutions to a sub-vocabulary of the representation language. This target language is referred to as the production field that defines the properties enforced upon the derived—so called characteristic—theorems.

Definition 2.8 (Production Field [Ino91]). A production field $\mathcal{P}$ is a pair $\langle \mathfrak{L}_\mathcal{P}, Cond \rangle$ where $\mathfrak{L}_\mathcal{P}$ is a subset of $L^{\pm}$, while *Cond* denotes the characteristic literals and defines a condition to be satisfied. If *Cond* is not given, the production field is denoted $\langle \mathfrak{L}_\mathcal{P} \rangle$.

$\mathfrak{L}_\mathcal{P}$ defines the limitations enforced upon the form of literals found in the derived consequences, while *Cond* characterizes general conditions of the solution, e.g., consequences with a certain cardinality. We say a clause C *belongs* to a production field, in case all literals in C are a subset of $\mathfrak{L}_\mathcal{P}$ and C fulfills the condition *Cond* [Ino91].

Definition 2.9 (Stable Production Field [Ino91]). A production field $\mathcal{P}$ is stable if for any two clauses C_i and C_j where $C_i \subseteq C_j$[8], it holds that if C_j *belongs* to $\mathcal{P}$ then also C_j *belongs* to $\mathcal{P}$.

[7] Of course, $\mathcal{T} \cup \neg\mathcal{M}^*$ have to be consistent.
[8] A clause C subsumes a clause C' if every literal in C occurs in C' [Ino91].

Definition 2.10 (Characteristic Clause [Ino91]). A clause C of Ψ is a characteristic clause w.r.t. to a production field $\mathcal{P}$ s.t. $\Psi \models C$ and is subset minimal. That is, a characteristic clause of Ψ is a prime implicate of Ψ belonging to $\mathcal{P}$.

Example 2.6

Let Ψ be $(a \rightarrow b) \wedge (\neg a \rightarrow b)$ and $\mathcal{P} =< \{b, \neg a\},$ of size *1* $>$, i.e., the target language allows clauses containing exactly 1 literal either b or $\neg a$. Then b is the only characteristic clause.

The prime implicates of a formula Ψ are equivalent to the main consequences, since they both define the logically strongest clauses that are implied by Ψ. Hence, we can utilize consequence finding for abductive reasoning, i.e., by defining the production field in such a way that only hypotheses can be part of the consequences. Thus, abductive explanations are computed deductively, in particular the theorem-proving procedure derives the prime implicates instead of the empty clause [Ino02].

Skip-Filtered Kernel-Resolution (SFK-resolution) [Val99] is a generalization of Tison's prime implicate algorithm. Del Val [Val99] uses this idea and splits each clause C into two set of literals; a *kernel* $k(C)$ denoting the "usable" literals and *skip* $s(C)$ the literals not resolved upon. Hence, each clause C is a pair $\langle s(C), k(C)\rangle$. A kernel deduction proof is then a resolution in which every literal is resolved upon the clause's *kernel*. Given a resolution step, the resolvent is split into *skip* and *kernel* based on the literal l resolved upon; i.e., each literal occurring before l is skipped, while each literal later in the ordering remains in the *kernel*. SFK-resolution is then a restriction of kernel resolution limiting the derived consequences to the target language.

***Example 2.7* [Mar00]**

Let Ψ be $(a \vee b) \wedge (a \vee c \vee d) \wedge (\neg b \vee \neg c)$, $\mathcal{P} =< \{a\} >$, and given an ordering $a < b < c < d$. The following sequence is an SFK-resolution proof:

1.	$\langle \emptyset, \{a, b\}\rangle$	input clause
2.	$\langle \emptyset, \{\neg b, \neg c\}\rangle$	input clause
3.	$\langle \{a\}, \{\neg c\}\rangle$	resolution of 1. and 2. upon b
4.	$\langle \emptyset, \{a, c, d\}\rangle$	input clause
5.	$\langle \{a\}, \{d\}\rangle$	resolution of 3. and 4. upon c

Each input clause C has $s(C) = \emptyset$ and $k(C) = C$. The first two input clauses can be resolved upon b, resulting in a new clause $\{a, \neg c\}$, however, since a is smaller than the literal b resolved upon it is added to the *skip* set, while $\neg c$ remains in the *kernel*. This clause's *kernel* can then be used for resolution with the input clause $\{a, c, d\}$ resulting in the clause in $5.$. At the end the *skip* contains the consequence a.

Simon and Del Val [SDV01] show an efficient implementation of kernel resolution based on bucket elimination and Zero Suppressed Binary Decision Diagrams (ZBDDs). ZBDDs provide a compact encoding and allow to process large clause sets based on resolution and subsumption. Their method exploits in addition to ZBDDs a variant of resolution called multiresolution that may be applied directly on the ZBDD representation. By specifying all allowed clauses, the authors only find implicates over a restricted language. Another consequence finding method is Finger and Genesereth's [FG85] RESIDUE procedure. It

generates deductive solutions for design synthesis based on the assumable variables, i.e., *residues*, left behind on resolution proofs.

Proof-Tree Completion

Besides using prime implicates or consequences for generating explanations, we can use another formalization often referred to as proof-tree completion [McI98]. Recall Definition 2.3, a set $\Sigma \subseteq \mathcal{H}$ is an explanation if $\mathcal{T} \cup \Sigma \models \mathcal{M}^*$ and Σ is consistent with the background theory. In consequence finding, we have rewritten the condition to $\mathcal{T} \cup \neg\mathcal{M}^* \models \neg\Sigma$ and obtained the explanations deductively. $\mathcal{T} \cup \Sigma \models \mathcal{M}^*$ is also equivalent to $\mathcal{T} \cup \Sigma \cup \neg\mathcal{M}^* \models \bot$. In this case, abductive explanations are derived as the refutations based on the theory and the negation of the observations. The procedure uses linear resolution trying to derive $\bot$ and has been first proposed by Pople [Pop73] and later also by Cox and Pietrzykowski [CP86]. In essence, one tries to prove the observations and the parts associated with failed branches of a proof-tree are the candidate explanations. By using linear resolution the method resolves the negated observations with the background theory, this leads to dead-ends that could be completed given the explanations. Theorist [Poo+87] also uses linear resolution to generate hypotheses, but instead of deriving dead ends the background theory is augmented with abducibles and the procedure records the hypotheses used to derive $\bot$ [McI98]. Given that linear resolution is proof-finding complete, these methods are based on a sound procedure.

2.2.2 Abduction by Set-covering

Peng and Reggia [PR90] have developed the parsimonious set covering theory as a formal framework for abductive reasoning. Their approach relies on an associative network of causal connections between disorders and manifestations. That is, the background theory $\mathcal{T}$ describes the causal relations between $\mathcal{H}$ and $\mathcal{M}$. Within $\mathcal{T}$ there exists a relation between a disorder or hypothesis h_i and a manifestation m_j whenever a disorder might cause a manifestations. Considering the logic-based abduction problem definition, there exists a relation between a h_i and m_j whenever there is a clause $h_i \rightarrow m_j$ contained within the theory. In its simplest form, the causal network consists of two layers of entities, i.e., hypotheses and manifestations. Peng and Reggia [PR90] refer to this type of abduction problem as a *bipartite problem*. We focus on these abduction problems.

Definition 2.11 (Set Cover Diagnosis Problem [PR90]). A diagnostic problem is a 4-tuple $\langle \mathcal{H}, \mathcal{M}, \mathcal{T}, \mathcal{M}^* \rangle$, where $\mathcal{H}$ is a finite, non-empty set of objects called hypotheses, $\mathcal{M}$ is a finite, non-empty set of objects called manifestations, $\mathcal{T} \subseteq \mathcal{H} \times \mathcal{M}$ is a relation with $\mathit{domain}(\mathcal{T}) = \mathcal{H}$ and $\mathit{range}(\mathcal{T}) = \mathcal{M}$ called causation, and $\mathcal{M}^* \subseteq \mathcal{M}$ is said to be the manifestations present.

A set of disorder $\Sigma \subseteq \mathcal{H}$ is an abductive explanation if Σ covers $\mathcal{M}^*$. In order to characterize a covering relation within this framework, we define for every hypothesis h_i the set $\mathit{effects}(h_i)$ and for every manifestation m_j the set $\mathit{causes}(m_j)$ [PR90]:

- $\mathit{effects}(h_i) = \{m_j | < h_i, m_j > \in \mathcal{T}\}$, i.e., the set of objects directly caused by h_i
- $\mathit{causes}(m_j) = \{h_i | < h_i, m_j > \in \mathcal{T}\}$, i.e., the set of objects which can directly cause m_j

Thus, for any subset of disorders $\mathcal{H}'$, we can determine the objects directly caused by it as

$$\mathit{effects}(\mathcal{H}') = \bigcup_{h_i \in \mathcal{H}'} \mathit{effects}(h_i) \ .$$

Along similar lines, we can observe that for any $\mathcal{M}' \subseteq \mathcal{M}$

$$\mathit{causes}(\mathcal{M}') = \bigcup_{m_j \in \mathcal{M}'} \mathit{causes}(m_j) \ .$$

Definition 2.12 (Cover [PR90]). A set $\mathcal{H}' \subseteq \mathcal{H}$ is said to cover $\mathcal{M}' \subseteq \mathcal{M}$ iff $\mathcal{M}' \subseteq$ *effects*$(\mathcal{H}')$.

A cover relation exists between a disorder and a manifestation whenever the latter is causally inferred from the former. While minimality is not a necessary condition for a cover in this first definition, Peng and Reggia [PR90] provide different parsimonious criteria, such as cardinality minimality (*minimum cover*) or subset minimality (*irredundant cover*). Since we are interested in subset minimal explanations, we will refer to *irredundant covers* as *minimal covers*.

Definition 2.13 (Minimal Cover [PR90]). A cover $\mathcal{H}'$ is said to be a minimal cover for a set of manifestation $\mathcal{M}'$ iff there exists no $\mathcal{H}'' \subseteq \mathcal{H}'$ such that $\mathcal{H}''$ is also a cover for $\mathcal{M}'$.

Definition 2.14 (Set Cover Explanation [PR90]). Given an abduction problem, a set $\Sigma \subseteq \mathcal{H}$ is said to be a solution to the abduction problem iff Σ is a minimal cover for $\mathcal{M}^*$.

Example 2.8

Assume the following abduction problem with $\mathcal{T}$ given by the causal associate network in Figure 2.4, $\mathcal{H} = \{H_1, H_2, H_3, H_4, H_5, H_6, H_7\}$, $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$, and $\mathcal{M}^* = \{m_1, m_2, m_4\}$. For example, *effects*$(H_1) = \{m_1, m_2\}$, *causes*$(m_1) = \{H_1, H_2, H_3, H_4\}$, *effects*$(\{H_3, H_7\}) = \{m_1, m_4\}$, and *causes*$(\{m_3, m_4\}) = \{H_4, H_5, H_6, H_7\}$.

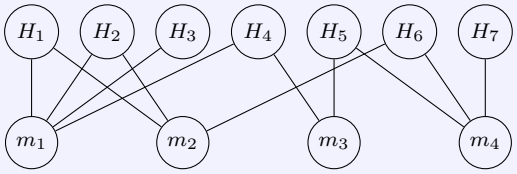


Figure 2.4: Causal associative network [PR90].

Considering $\mathcal{M}^*$, for instance, the set $\{H_1, H_2, H_6, H_7\}$ covers the current manifestations, since *effects*$(\{H_1, H_2, H_6, H_7\}) = \{m_1, m_2, m_4\}$. However, this cover is not parsimonious, since for example $\{H_1, H_6\}$ is sufficient to explain $\mathcal{M}^*$. Given this abduction problem there are eight minimal covers, i.e., explanations: $\{H_1, H_5\}$,$\{H_1, H_6\}$,$\{H_1, H_7\}$,$\{H_2, H_5\}$,$\{H_2, H_6\}$,$\{H_2, H_7\}$,$\{H_3, H_6\}$, and $\{H_4, H_6\}$.

To derive explanations for abduction problems based on bipartite networks, the authors present a sequential construction of solutions for one manifestation at a time. They propose a compact representation of explanations as *generators* and define a set of operations on them to derive solutions.

Definition 2.15 (Generator [PR90]). Let $g_1, g_2, \ldots, g_n$ be non-empty pairwise disjoint subsets of $\mathcal{H}$. Then $G_I = \{g_1, g_2, \ldots, g_n\}$ is a generator. The class generated by G_I is defined as $[G_I] = \{\{H_1, H_2, \ldots, H_n\} | H_i \in g_i, 1 \leq i \leq n\}$.

Definition 2.16 (Generator-Set [PR90]). $G = \{G_1, G_2, \ldots, G_N\}$ is a generator-set if each $G_I \in G$ is a generator and $[G_I] \cap [G_J] = \emptyset$ for $I \neq J$. The class generated by G is $[G] = \bigcup_{I=1}^{N} [G_I]$.

A generator can be interpreted as the Cartesian Product based on unordered sets, i.e., a generator G_I characterizes all possible sets of elements which can be formed by taking one element of each $g_i \in G_I$. A generator-set represents the classes created by each of its members, such that there is no overlap between the classes.

Example 2.9

Assume two generators:

$$G_1 = (\{H_1, H_2\}, \{H_5, H_6, H_7\}) \quad \text{and} \quad G_2 = (\{H_3, H_4\}, \{H_8\}) \; .$$

G_1 implicitly represents six explanations, i.e., $\{H_1, H_5\}, \{H_1, H_6\}, \{H_1, H_7\}, \{H_2, H_5\}$, $\{H_2, H_6\}$, and $\{H_2, H_7\}$.

Combining G_1 and G_2 to $G = \{G_1, G_2\}$, we can encode eight explanations: $[G] = \{\{H_1, H_5\}, \{H_1, H_6\}, \{H_1, H_7\}, \{H_2, H_5\}, \{H_2, H_6\}, \{H_2, H_7\}, \{H_3, H_8\}, \{H_4, H_8\}\}$.

Peng and Reggia present Algorithm BIPARTITE that incrementally computes the explanations for a set of observations, considering one manifestation m_{new} at a time. By manipulating generator-sets, all minimal covers can be extracted using three operations: division `div`, division residual `res`, and the augmented residual of a division `augres`. `Div` returns a generator-set characterizing all explanations of all previously considered observations, which also cover the newly added manifestation m_{new}. The residual of a division `res` represents all explanations of the previously considered manifestations, which do not cover m_{new} and `augres` contains in addition to the residual also elements from *causes*(m_{new}) ensuring the generator-set covers m_{new}. Once all observations have been considered, the computation has finished and the returned generator-set represents all explanations to the diagnosis problem. Due to its iterative nature, BIPARTITE is suitable for open problems, i.e., only a subset of observations is available at a time.

Algorithm 2.3: **BIPARTITE [PR90]**

Require: $\mathcal{H}$: the set of hypotheses, $\mathcal{M}$: the set of manifestations, $\mathcal{T}$: the relation between hypotheses and manifestations, M^*: the set of observations
Ensure: G generator-set, representing the solutions to the abduction problem

1: $G \leftarrow \emptyset$
2: **while** $M^* \neq \emptyset$ **do** ▷ Incrementally update G by *causes*(m) for $m \in M^*$
3: $m_{new} \leftarrow$ `nextManifestation` ▷ Remove the next manifestation from M^*
4: $F \leftarrow \texttt{div}(G, \textit{causes}(m_{new}))$ ▷ Explanations of G covering m_{new}
5: $Q \leftarrow \texttt{augres}(G, \textit{causes}(m_{new}))$ ▷ Explanations covering $G \cup m_{new}$
6: $G \leftarrow F \cup \texttt{res}(Q, F)$ ▷ Removes all redundant and duplicated covers
7: **end while**
8: **return** G

Example 2.8 (cont.)

BIPARTITE sequentially considers each manifestation in $\mathcal{M}^*$; for the initial $m_{new} = m_1$, G simply contains all *causes* of m_1, i.e., $G = \{H_1, H_2, H_3, H_4\}$. Taking into account the second observation, i.e., $m_{new} = m_2$ we have to compute F, the division between the generator and the causes of m_2, and Q, the augmented residual covering m_2, to derive the generator-set representing the minimal covers. We continue until all manifestations have been considered.

m_{new}	$causes(m_{new})$	F	Q	G
m_1	$\{H_1, H_2, H_3, H_4\}$	$\emptyset$	$\{(\{H_1, H_2, H_3, H_4\})\}$	$\{(\{H_1, H_2, H_3, H_4\})\}$
m_2	$\{H_1, H_2, H_6\}$	$\{(\{H_1, H_2\})\}$	$\{(\{H_3, H_4\}, \{H_6\})\}$	$\{(\{H_1, H_2\}),$ $(\{H_3, H_4\}, \{H_6\})\}$
m_4	$\{H_5, H_6, H_7\}$	$\{(\{H_3, H_4\},$ $\{H_6\})\}$	$\{(\{H_1, H_2\},$ $\{H_5, H_6, H_7\})\}$	$\{(\{H_1, H_2\},$ $\{H_5, H_6, H_7\}),$ $(\{H_3, H_4\}, \{H_6\})\}$

Once there are no more manifestations left to process, the last generator G contains our solutions, i.e., $\Sigma \equiv [G] = \{\{H_1, H_5\}, \{H_1, H_6\}, \{H_1, H_7\}, \{H_2, H_5\}, \{H_2, H_6\}, \{H_2, H_7\}, \{H_3, H_6\}, \{H_4, H_6\}\}$.

Peng and Reggia [PR90] provide an extension of their parsimonious covering theory, which handles intermediate, states between manifestations and hypotheses. Given these so called *layered problems*, which are restricted to cycleless networks, indirect causation has to be accounted for. Another extension utilizes probabilistic causal models to be able to rank explanations. Recall that the associative networks do not guarantee the occurrence of a symptom given a disorder. To account for this uncertainty, the authors assign to each connection a causal strength value. In addition, assuming statistical independence of hypotheses, given a-priori disorder probabilities allow to represent the quality of a solution. Several refinements to the basic theory have been proposed such as the improvement of models with additional knowledge or the inclusion of more complex covering relations [BS02].

Besides using algorithm BIPARTITE, Peng and Reggia [PR89] propose a two-layered connectionist architecture, where the sets of disorders and manifestations are neurons and $\mathcal{T}$ describes the set of links between the layers. Each node has a constant and a dynamically changing value for its activation level. For each $h_i \in \mathcal{H}$ the constant value is the hypothesis' prior probability, while the activation level ranges from 0 to 1 and indicates whether the disorder is confirmed or not. For $m_j \in M$ the constant value is either 1 if the manifestation is observed or 0 otherwise. Its dynamic value depends on the activation of all the disorders causing said manifestation. A neuron updates its activation level when it receives information from its neighboring nodes and subsequently sends its own activation level to its neighbors. In particular, the hypothesis nodes compete for the activity level of their connected manifestation nodes. Once the model converges to a set of explanations nodes, i.e., the activation levels of the disorder nodes are close to 1 or 0 respectively, then the fully activated neurons compose the solution. Ayeb, Wang, and Ge [Aye+98] define a unified model for abduction problems also relying on a neural architecture. Their bipartite network consists of observation as well as hypothesis nodes, which are connected via weighted links. The weight between a disorder and a manifestations represents the belief in the presence of the hypothesis given the observation. Other work using neural networks and set-covering diagnosis has been proposed by Abdelbar, Andrews, and Wunsch [Abd+03]. The authors utilize high-order neural networks for cost-based abduction seeking minimal cost solutions. Their network uses penalty logic, an extension of propositional logic, which associates real-valued penalties with well-formed formulas.

2.2.3 Probabilistic Abduction

The drawback of the previously described approaches is that there is possibly a large number of minimal solutions. Probabilistic methods restrict the admissible explanations by searching for the most probable ones. Besides structural information this framework

also accounts for probabilistic knowledge about causes and effects. In particular, the prior probabilities of hypotheses as well as conditional probabilities between elements of $\mathcal{H}$ and $\mathcal{M}$ are utilized [EG95]. For instance, the simple parsimonious set-covering framework has been extended such that a-priori probabilities of disorders as well as the strengths, i.e., frequencies, of causal connections are incorporated. Based on this information, the relative likelihood of a diagnosis given the set of observations can be calculated. In essence, abductive inference in the probabilistic framework can be viewed as a combinatorial optimization problem: the best solution $\Sigma \subseteq \mathcal{H}$ maximizes the a-posteriori probability $P(\Sigma|\mathcal{M}^*)$. Usually the problem solver computes the k most probable explanations given the evidence.

Bayesian Belief Networks (BBNs) were first introduced by Pearl [Pea88] and are Directed Acyclic Graphs (DAGs) in which nodes correspond to random variables and edges characterize conditional dependencies. Each root node contains a prior probability, while each non-root node has a conditional probability based on the node's parents. A BBN represents the joint probability distribution $P(V) = \prod_{i=1}^{n} P(V_i|pa(V_i))$ over its variables, where $pa(V_i)$ denotes the parents of V_i in the graph. Probabilities of interest can be computed by Bayes' theorem. In the context of probabilistic reasoning, abduction corresponds to the problem of finding the joint value assignment to a set of variables in the network which has the highest a-posterior probability given the observed values of other variables in the network. It is known as Most probable explanation (MPE) or Maximum a posteriori assignment problem (MAP). MAP is the search for the most probable instantiation of a set of unobserved variables in a Bayesian network given some evidence. MPE, which is a special case of MAP, is the search for the most probable instantiation of a set of unobserved variables given a particular instantiation of the remaining variables [PD01].

While there are exact methods to derive the most probable explanation, there are various approximation techniques. For example, Fortier, Sheppard, Pillai [For+13] propose a swarm-based approximation approach to compute k-MPE of a Bayesian network. An overlapping swarm intelligence algorithm is utilized in which a particle swarm is associated with each non-evidence node in the network. Charniak and Shimony [CS94] use cost-based abduction to find the "best" explanation as the one with a minimal cost proof taking into account the costs of assumptions and rules. Given their cost-based abduction they define a probabilistic semantic for the costs such that they can show the equivalence to finding a maximum-a-posteriori assignment over a belief network. A similar idea has been followed by Hobbs et al. [Hob+93] in weighted abduction for natural language interpretation. Each literal in an antecedence is assigned a cost representing the effort of proving it instead of assuming it and a function favors proofs with fewer assumptions. Ovchinnikova, Gordon, and Hobbs [Ovc+13] show how certain features of weighted abduction can be represented in a formal probabilistic framework based on Bayesian Networks.

2.3 Abductive Model-Based Diagnosis

The concepts of model-based reasoning emerged from the limitations of expert systems. Expert systems exploit knowledge stemming from experience within a specific context for problem solving. This type of "shallow knowledge" contrasts the "deep knowledge" from first principle (i.e., information on physical component behavior) that model-based diagnosis is built upon [Str08]. Model-based reasoning was envisioned by Davis [DH88] as a means to determine faults by using structural and behavioral information. It fosters the idea of reusing knowledge by relying on a formalization of components. Given a component library various physical systems can be represented by combining the models of their constituting parts.

The system model together with a set of observed symptoms can be exploited to obtain diagnostic hypotheses for the observations. Reiter [Rei87] proposed a formal definition of the problem based on a component-oriented system description in first order logic characterizing the correct system behavior. His approach uses the notion that, in the presence of a fault a discrepancy between the fault free and the observed behavior manifests itself as conflicting assumptions of components' health states. A diagnosis then is a set of abnormality assumptions about the system such that the observations and assumptions are consistent. The set of all root causes are obtained as hitting sets on the derived conflicts. At the same time de Kleer and Williams [DKW87] developed the General Diagnosis Engine (GDE) for detecting and identifying multiple faults, using an Assumption-based Truth Maintenance System (ATMS) to uncover conflicts and on that basis compute diagnoses. In addition, a technique for selecting the next best measurement was introduced. Further developments include improvements upon the GDE itself [FK88] and the introduction of fault modes [SD89] and physical impossibilities[9] [Fri+90b]. Since the fault detection is achieved through finding inconsistencies, the methodology is known as consistency- based diagnosis. Of course the model-based approach relies on certain assumptions, such as reliability of the system description and observations or that a discrepancy between the measured and predicted behavior does not stem from a design-error, but a fault in the system [FB+97].

Model-based diagnosis can look back at decades of research with various advances in the computation of diagnosis, modeling strategies, and applications to real world problems. In contrast the variants deriving inconsistencies and then diagnoses, Fröhlich and Nejdl [FN97] directly manipulate the logical model. Most recently, Metodi et al. [Met+14] present a SAT encoding for consistency-based diagnosis. The system description is compiled into a Boolean formula, such that the formula's satisfying assignments correspond to the solutions of the diagnosis problem. Based on the encoding, a SAT solver directly computes the diagnoses. In order to improve the solver's performance, the authors utilize several preprocessing techniques. An empirical comparison of their approach to other model-based diagnosis algorithms indicates that their SAT encoding yields performance benefits. Contrasting these results, Feldman et al. [Fel+10b] propose a translation to Max-SAT which could not outperform a stochastic model-based diagnosis algorithm. In Nica and Wotawa [NW12] the authors present an algorithm which ties constraint solving to diagnosis, thus renders the detection of inconsistencies and subsequent hitting set computation unnecessary. Another direct approach by Felfernig and Schubert [FS10] computes minimal diagnoses for over-constrained problems by finding the sets of constraints to be relaxed in order to restore consistency. Nayak and Williams [NW97] introduced an incremental approach that allows for fast context switching. Their diagnosis and reconfiguration system for space autonomous agents, called Livingston, was used in NASA's Deep Space One [WN96]. Besides spacecraft systems, model-based reasoning techniques have been applied to on-board diagnosis in the automotive industry [Str+96], configuration and reconfiguration systems [Fel+04], and software debugging [Fri+99].

In contrast, abductive model-based diagnosis is based on the notion of logical entailment [Poo+87]. A set of abnormality assumptions entailing the observations constitutes an abductive diagnosis or explanation. To derive causes for observed anomalies utilizing this type of inference, the abductive model-based diagnosis approach depends on a model representing the links between faults and their manifestations. Even though consistency-based and abductive diagnosis are based on different reasoning techniques, Console et al. [Con+91] showed the close relation between both variations. While the consistency-based method is more popular, there are applications of abductive model-based diagnosis. For instance,

[9] Instead of representing various fault modes, Friedrich et al. [Fri+90b] characterize situations that are impossible under all modes, e.g., a bulb without power is never lit.

Portinale, Magro, and Torasso's [Por+04] ADAPtER combines model-based and case-based reasoning for diagnosis. Given a new diagnosis problem, the case memory is searched for an appropriate solution. If the solution is not consistent and complete given the new diagnosis problem, it is adapted using the behavioral model and abductive diagnosis. Wotawa, Rodriguez-Roda, and Comas [Wot09; Wot+09] apply abductive model-based diagnosis to improve environmental decision support systems. Their approach is based on using the ATMS as a consequence finding procedure.

Propositional Horn Clause Abduction

Originally, abductive model-based approaches operate on formalizations of the faulty system behavior. Such failure knowledge can usually be expressed as Horn theories [CT91]. Besides being a suitable representation for diagnostic models, the complexity results for this subset of logic are less daunting: in case of general propositional theories and clausal theories, abduction is located in the second level of the polynomial complexity hierarchy, while for Horn clauses the complexity is mitigated to the first level [EG95]. Hence, we focus on a less expressive modeling language, in our case Horn clause models. A Horn clause is defined as a disjunction of literals featuring at most one positive literal and can be described by a rule, e.g., $\{\neg a_1, \ldots, \neg a_n, a_{n+1}\}$ can be written as $a_1 \wedge \ldots \wedge a_n \rightarrow a_{n+1}$. Similar to Friedrich, Gottlob, and Nejdl [Fri+90a][10], we define a *KB* representing the abductive diagnosis model in the context of propositional Horn clause abduction.

Definition 2.17 (Knowledge Base (*KB*) [Wot+09]). A knowledge base (*KB*) is a tuple (*A*,*Hyp*,*Th*) where *A* denotes the set of propositional variables, $Hyp \subseteq A$ the set of hypotheses, and *Th* the set of Horn clause sentences over *A*.

A hypothesis, also referred to as an assumption, is a propositional variable for which we can presume a certain truth value. Hypotheses are the propositions which can be part of a diagnosis, i.e., are abducible, while the Horn theory depicts the relationships between the variables, i.e., represents the system description.

***Example 2.10* (adapted from [Tag05])**

Consider an ATM system. Assuming an ATM dispenses cash given a valid debit card, the input of the correct pin, and the fact that the balance on the account linked to the debit card is positive. Yet, the safe and cash-dispensing mechanism itself may observe several issues. In a simplified world, the ATM may not return any money in case it has run out of cash or the machine is jammed. Even in cases where the machine dispenses currency, the amount delivered may differentiate from the one requested in case bills are stuck together and the integrated sensor does not correctly evaluate the thickness of the notes or the denominations have been stacked in the wrong tray. Hence, the account is charged with an inaccurate total. In case a power interruption occurs while the withdrawal request is being processed, a transaction failure arises that leads to an incorrect charge on the debit

[10] Friedrich, Gottlob, and Nejdl [Fri+90a] investigate definite Horn clauses, i.e., Horn clauses featuring exactly one positive literal.

card. Given the description we can create the following Horn theory representation of the circumstances[a]:

$$Th = \left\{ \begin{array}{c} Out\_of\_cash \rightarrow no\_cash, Machine\_jam \rightarrow no\_cash, \\ Sensor\_fault \wedge Bills\_stuck\_together \rightarrow incorrect\_amount, \\ Denominations\_incorrectly\_stacked \rightarrow incorrect\_amount, \\ incorrect\_amount \rightarrow incorrect\_charge, \\ Power\_failure \rightarrow transaction\_failure, transaction\_failure \rightarrow incorrect\_charge \end{array} \right\}$$

The set of propositional variables A comprises all hypotheses as well as propositions representing effects:

$$A = \left\{ \begin{array}{c} Out\_of\_cash, no\_cash, Machine\_jam, Sensor\_fault, \\ Bills\_stuck\_together, incorrect\_amount, Denominations\_incorrectly\_stacked, \\ incorrect\_charge, Power\_failure, transaction\_failure \end{array} \right\}$$

The faults we want to identify during diagnosis form the set of hypotheses.

$$Hyp = \left\{ \begin{array}{c} Out\_of\_cash, Machine\_jam, Sensor\_fault, \\ Denominations\_incorrectly\_stacked, Power\_failure \end{array} \right\}$$

[a] In this example, we use the convention often seen in the ATMS literature that assumptions start with a capitalized letter.

Definition 2.18 (Propositional Horn Clause Abduction Problem (PHCAP) [Wot+09]). Given a knowledge base (*A*,*Hyp*,*Th*) and a set of observations $Obs \subseteq A$ then the tuple (*A*,*Hyp*,*Th*,*Obs*) forms a propositional Horn clause abduction problem (*PHCAP*).

A diagnosis problem involves a *KB* as well as a set of observations for which the explanations are to be computed. In our context, these observables may only be a conjunction of propositions and not an arbitrary logical sentence. The solution to a *PHCAP* or diagnosis Δ is a set of hypotheses explaining the propositions in *Obs*, i.e., entailing them together with the theory *Th*. In other words, the observations are a logical consequence of the failure relations described in the theory and the determined explanation. An additional requirement is that only consistent diagnoses are permitted, thus solutions leading to a contradiction are disregarded. Imposing a parsimonious criterion on the solutions is a principle commonly used in diagnosis. From our practical point of view only subset minimal explanations are of interest.

Definition 2.19 (Diagnosis; Solution [Wot+09]). Given a *PHCAP* (*A*,*Hyp*,*Th*,*Obs*). A set $\Delta \subseteq Hyp$ is a solution if and only if $\Delta \cup Th \models Obs$ and $\Delta \cup Th \not\models \bot$. A solution Δ is parsimonious or minimal if and only if no set $\Delta' \subset \Delta$ is a solution. Δ-Set contains all solutions obtained from a *PHCAP*.

Example 2.10 (cont.)

Considering the *PHCAP* and assuming the ATM dispenses an incorrect currency amount, i.e., $Obs = \{incorrect\_amount\}$, we can derive two minimal explanations; either the sensor checking the thickness of the returned bills is broken in conjunction with a set of bills being stuck together ($\Delta_1 = \{Sensor\_fault, Bills\_stuck\_together\}$) or the teller has confused

the trays whilst loading the banknotes ($\Delta_2 = \{Denominations\_incorrectly\_stacked\}$), i.e., Δ-Set $= \{\{Sensor\_fault, Bills\_stuck\_together\}, \{Denominations\_incorrectly\_stacked\}\}$.

While abductive reasoning provides an intuitive approach for fault localization, there might be a large number of diagnoses that is exponential in the number of abducible atoms. Its computational complexity for general propositional theories is located within the second level of the polynomial hierarchy [EG95]. Focusing on a less expressive modeling languages, such as in our case Horn clause models, reduces the complexity. Friedrich et al. [Fri+90a] focus on definite Horn theories and showed that while checking whether a solution exists is polynomial, deciding the relevance of a hypothesis in the context of subset minimal explanations is NP-complete. Eiter and Makino [EM03] give a broader set of results, showing that for Horn theories given a single positive literal as observation the computation of all explanations is possible in polynomial total-time[11]. The same holds for a positive term, i.e., the conjunction representing the observation only contains positive literals. For a single negative literal or general term, the problem is not polynomial total-time solvable.

2.3.1 Abduction with the Assumption-Based Truth Maintenance System

There are various techniques for computing abductive explanations, as mentioned in the previous section. We want to present one well-known approach that has first been introduced in the context of consistency-based diagnoses; i.e., the ATMS [DK86a]. The ATMS functions as an abductive reasoning engine for propositional Horn theories and can be understood as a propositional consequence finding procedure returning for a given query a logical consequence of the theory [McI98; Lev89]. It interacts with a problem solver. The problem solver sends inferences to the ATMS and the ATMS decides on the belief of the data. Internally an And-or-graph of the theory is constructed, where the contradiction and propositions are vertices and the directed edges between them are dependent on the implications within the theory. Based on this representation the ATMS creates and updates a label Λ for each node n such that the label records which hypotheses support the vertex. More specifically, the label comprises a set of environments. An environment is a set of assumptions that allows to infer the corresponding node. Since hypotheses are justified by themselves, the label of an assumption consists of one environment having the assumption itself as the only element. By maintaining the information within the labels, the ATMS stores all entailment relations between variables within the underlying theory. To ensure that labels are consistent, the ATMS retains a `NOGOOD` node recording contradicting assumptions. These inconsistencies as well as their supersets are not viable within the vertices' labels. de Kleer [DK86c] describes a procedure for computing consistent, sound, complete, and subset minimal labels. The algorithm incrementally updates the nodes' labels whenever a new Horn clause is added. Many improvements to de Kleer's original method for computing solutions based on ATMS exist, for instance, to account for uncertainty [Dub+91] or more expressive logical representations [DK86b].

Due to the fact that the ATMS labels comprise all hypotheses justifying a corresponding node and are consistent, sound, complete, as well as minimal, the ATMS already derives parsimonious abductive explanations. In particular, the ATMS returns a conjunction of literals Σ drawn from *Hyp* such that $\neg\Sigma \vee Obs$ is a prime implicate of *Th* [McI98]. Algorithm ABDUCTIVEEXPLANATIONS [Wot+09] computes all minimal explanations given a *PHCAP*. First, the theory *Th* is supplied to the ATMS by adding each Horn clause via the

[11] The diagnoses can be generated in time polynomial in the combined size of the input and entire output [Fri+90a].

method ATMS_PROPAGATE. Given a clause, ATMS_PROPAGATE incrementally computes the necessary label updates within the And-or-graph. Once the information of the entire theory has been distributed, we utilize the ATMS to obtain diagnoses. Explanations for a single symptom can be extracted directly from the proposition's vertex label, since it stores the assumptions from which the node can be inferred from without leading to a contradiction. In case of multiple observations, an additional rule is applied to the ATMS, such that a conjunction of the observations is present in the body and a new propositional variable forms the head, i.e., $\bigwedge_{o \in Obs} o \rightarrow ex$, where ex is not yet contained within A. Every set of hypotheses included within the label of ex then constitutes a minimal consistent explanation. As the set of hypotheses is finite, computing abductive explanations via the ATMS is guaranteed to terminate.

Algorithm 2.4: ABDUCTIVEEXPLANATIONS **(based on [Wot+09])**

Require: A: the set of propositional variables, Hyp: the set of hypotheses, Th: the Horn theory, Obs: the set of observations
Ensure: Δ-Set: set of minimal explanations
  for all $hc \in Th$ **do**
    ATMS_PROPAGATE(hc) ▷ Adding all Horn clauses of the theory to the ATMS
  end for
  if $|Obs| > 1$ **then**
    ATMS_PROPAGATE($\bigwedge_{o \in Obs} o \rightarrow ex$) ▷ ex is a new proposition, i.e., $ex \notin A$
    Δ-Set $\leftarrow \Lambda(ex)$ ▷ Retrieving the label of node ex containing the explanations for Obs
  else
    Δ-Set $\leftarrow \Lambda(o)$ ▷ Retrieving the label of the observation node $o \in Obs$
  end if
  return Δ-Set

Example 2.10 (cont.)

To derive the abductive explanations for Example 2.10, we exploit the ATMS as follows: Figure 2.5 depicts the And-or-graph given the first two Horn clauses. Each node is depicted with its name and underneath its label Λ is shown in red. For the assumption node *Out_of_cash*, for instance, we can observe that the hypothesis justifies itself, while *no_cash*'s label value is determined by its antecedent nodes. In Figure 2.6 we see the graph representing all sentences of Th. Given $Obs = \{incorrect\_amount\}$, we can simply extract the minimal explanations from the corresponding node's label $\Lambda(incorrect\_amount)$.

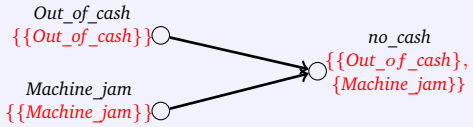


Figure 2.5: And-or-graph considering the first two clauses of Th.

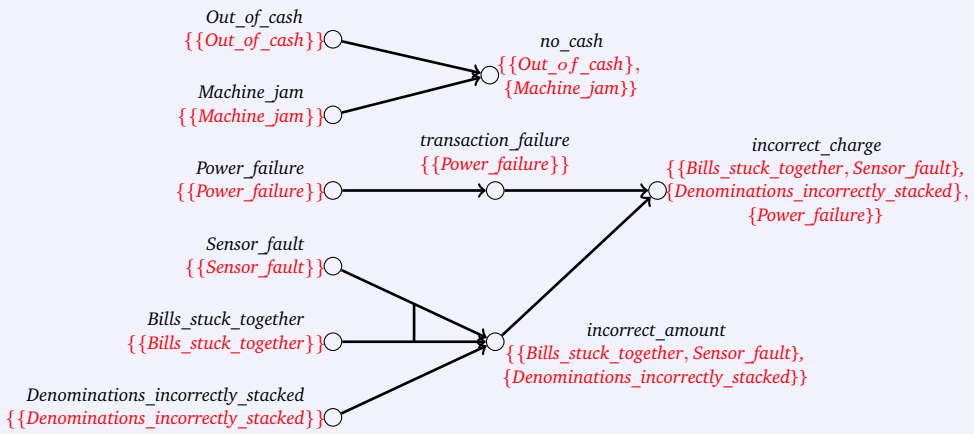


Figure 2.6: And-or-graph after all clauses have been considered.

2.3.2 Reformulating the PHCAP

In logic-based abduction, we rely on the notion of entailment; an abductive explanation or diagnosis is composed of a set of abducible propositions, i.e., fragments permitted to be part of a solution, entailing the observed symptoms together with the theory while not leading to a contradiction. Considering the definition of abduction, we can characterize two more approaches for computing solutions to a *PHCAP*: proof-tree completion [McI98] and consequence finding [Mar00], which we have already discussed previously in Section 2.2.1. Here, we present some definitions of diagnosis based on the *PHCAP* in these two problem reformulations.

Proof-Tree Completion

In proof-tree completion, abductive diagnosis is re-framed to the search for a refutation proof consisting of abducible propositions. By logical equivalence, we can reformulate the relation between a diagnosis, the background theory and the observations to $\Delta \cup Th \cup \{\neg Obs\} \models \bot$, where $\{\neg Obs\}$ is the disjunction containing a negation of each observation, i.e., $\bigvee_{o_i \in Obs} \neg o_i$. Thus, we can rewrite the theory in such a way that a contradiction arises given the negated observations. To extract explanations, the derived conflicts have to be propositions from *Hyp* and again no inconsistency may arise from the diagnoses.

Definition 2.20 (Conflict). Given a *PHCAP*(*A*,*Hyp*,*Th*,*Obs*) a conflict $\mathcal{CO}$ is a set $\{c_1, \ldots, c_k\} \subseteq Hyp$ such that $\mathcal{CO} \cup Th \cup \{\neg o_1 \vee \ldots \vee \neg o_n\}$ is inconsistent, where $\{o_1, \ldots, o_n\} = Obs$. A conflict is minimal iff there is no conflict $\mathcal{CO}'$ such that $\mathcal{CO}' \subset \mathcal{CO}$.

Definition 2.21 (Conflict-Driven Diagnosis). Given a *PHCAP*(*A*,*Hyp*,*Th*,*Obs*). A set Δ is a solution iff Δ is a conflict and $\Delta \cup Th \not\models \bot$. A solution is parsimonious, iff Δ is a minimal conflict.

Example 2.10 (cont.)

Given our *PHCAP* from before, we assume that we received the correct withdrawal, i.e., $\neg$*incorrect_amount* and that all our hypotheses are true, e.g., the machine has run out of cash, the machine is jammed, and there is a power failure etc. This leads to conflicts: for instance, according to the theory given the teller has incorrectly stacked the ATM,

we must get an incorrect amount of cash; thus, $\{Denominations\_incorrectly\_stacked\} \cup Th \cup \{\neg incorrect\_amount\}$ is inconsistent. Extracting only the hypotheses, we retrieve the first minimal conflict $\mathcal{CO}_1 = \{Denominations\_incorrectly\_stacked\}$. Since the conflict is consistent with the theory *Th*, $\mathcal{CO}_1$ is a minimal diagnosis. The second parsimonious conflict and diagnosis can be extracted analogously.

Consequence Finding

By further rewriting the relation from Definition 2.19, we can construct $Th \cup \{\neg Obs\} \models \{\neg\Delta\}$, where $\{\neg\Delta\} = \bigvee_{\delta_j \in \Delta} \neg\delta_j$[12]. In this scenario, abductive explanations are obtained in a deductive manner by finding the logical consequences of the theory and the negated observations comprising negations of hypotheses.

Definition 2.22 (Consequence). Given a *PHCAP*(*A*,*Hyp*,*Th*,*Obs*) a consequence $\mathscr{C}$ is a set $\{c_1, \ldots, c_k\} \subseteq Hyp$ such that $Th \cup \{\neg o_1 \vee \ldots \vee \neg o_n\} \models \{\neg c_1 \vee \ldots \vee \neg c_k\}$, where $\{o_1, \ldots, o_n\} = Obs$. A consequence is minimal iff there is no consequence $\mathscr{C}'$ such that $\mathscr{C}' \subset \mathscr{C}$.

Definition 2.23 (Consequence-Finding Diagnosis). Given a *PHCAP* (*A*,*Hyp*,*Th*,*Obs*). A set Δ is a solution iff Δ is a consequence and $\Delta \cup Th \not\models \bot$. A solution is parsimonious, iff Δ is a minimal consequence.

Example 2.10 (cont.)

Again we state that we have obtained the correct total from the ATM, i.e., $\neg incorrect\_amount$. This means, however, that the bills have to be stacked in the correct trays given this observation since the relation in *Th* postulates that if the teller in fact did mis-distribute the denominations, then we should have received an incorrect withdrawal. Looking at the entailment relation, we can deduce that the negation of *Denominations_incorrectly_stacked* is a logical consequence of the theory and the observation, i.e., $Th \cup \{\neg incorrect\_amount\} \models \{\neg Denominations\_incorrectly\_stacked\}$. Hence, $\{Denominations\_incorrectly\_stacked\}$ is one of the consequences as defined in Definition 2.22. Since the consequence is further consistent with *Th*, $\{Denominations\_incorrectly\_stacked\}$ is a minimal fault diagnosis. Similarly, we can derive the second diagnosis.

[12] This requires that $Th \cup \{\neg Obs\} \not\models \bot$.

Part II

From Theory to Practice

3 A Process for Applying Model-Based Diagnosis in Industrial Applications

> *In theory, there is no difference between theory and practice. But in practice, there is.*
>
> — **Walter J. Savitch**[1]
> "*Pascal: An Introduction to the Art and Science of Programming*". 1984.

This chapter is based on the following publications:

- [KW15b] Roxane Koitz and Franz Wotawa. „From Theory to Practice: Model-Based Diagnosis in Industrial Applications“. In: *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society*. 2015, pp. 197–205
- [KW15c] Roxane Koitz and Franz Wotawa. „On the Computational Feasibility of Abductive Diagnosis for Practical Applications“. In: *IFAC-PapersOnLine* 48.21 (2015). 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, pp. 410–415
- [KW16b] Roxane Koitz and Franz Wotawa. „Integration of Failure Assessments into The Diagnostic Process“. In: *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society*. 2016, pp. 117–128

The process has been published in [KW15b] while the detailed analyses of the failure assessments and modeling approaches in Section 3.3 are taken from [KW16b]. In [KW15c], we present the complexity results for diagnosis based on Failure Modes and Effect Analyses and describe the first empirical evaluation. The second experiment was published in [KW15b].

3.1 Motivation

Accurate failure localization in technical systems is a topic of interest from an industrial as well as research point of view due to the increasing complexity and magnitude of systems. An extensive body of research has concerned itself with this subject ranging from fields such as Control Engineering to AI. As we have discussed, model-based diagnosis has been proposed as reasoning from first principle to avoid the disadvantages associated with rule-based systems. Even though model-based approaches build on solid theories, have been applied to various domains [WN96; Str+96; Wot+10], and continuous research improves their methods, a widespread adoption is yet to be observed. Certainly, the initial creation of the system descriptions suitable for fault identification poses an obstacle, the difficulties of integrating fault identification in current work processes, as well as the computational complexity associated with diagnostic reasoning [CD99]. Further, as noted by Milde et al. [Mil+00], the current industrial work processes have to be known in order to integrate model-based diagnosis successfully. To account for the complexity of systems, the

[1] Also attributed to Jan L. A. van de Snepscheut and Yogi Berra.

models have to be developed in relation to existing knowledge on products, such as design, construction, and failure information. Travé-Massuyès and Milne [TMM98] identified the unavailability of automatic modeling tools and the complexity of seamlessly combining model-based applications with present software as technical gaps still hindering a vast adoption in practice. Thus, several projects have been engaged in developing methods to integrate model-based reasoning into industrial processes [Fle+01; Mil+00].

An essential benefit promoted by the model-based diagnosis community has been the reuse of knowledge. To emphasize this potential and provide an economic additional benefit besides troubleshooting, research has focused on further tasks which can be performed with the initially generated diagnosis models. Struss, Malik, and Sachenbacher [Str+96] describe an automated approach combining diagnosis and the creation of FMEAs as well as repair manuals based on compositional qualitative models of the intended part behavior. An FMEA is a systematic analysis of possible component faults and the consequences said faults have on the system behavior and function [HW98]. FMEA as a reliability analysis tool is growing in importance as it has been established as a mandatory task in certain industries, especially for systems that require a detailed safety assessment [Cat+10]. Similar goals to the one's by Struss, Malik, and Sachenbacher [Str+96] have been pursued by Hawkins and Woollons [HW98], or Price and Taylor [PT02]. While the former two proposed methods for FMEA, the latter automatically generates fault trees from device knowledge by predicting the correct and erroneous behavior based on qualitative models.

In contrast, we propose a different approach to reuse knowledge: As certain reliability assessments, such as FMEA or Fault Tree Analysis (FTA) are common practice and even mandatory in certain industries [RH04], the information captured can be exploited for diagnostic reasoning. In particular, these types of analyses describe the relation of failures and their consequences on the system, thus, can form the basis for abductive diagnosis. For instance, considering the records of an FMEA; each row represents a clear causal dependency from a specific fault mode to a set of symptoms, i.e., it describes the system's response to a fault [Wot14]. Kuntz et al. [Kun+11] also stress the importance of reducing or eliminating the modeling effort in order to incorporate diagnosis in real world problems. Applied to avionics systems, they propose a method based on engineering documents, such as FMEAs and other records describing the connection of components, to build a system model for diagnosis. In contrast to our work, they transform the resulting description, which encompasses the relation between component faults and their monitoring observation, to a finite state machine. Based on this representation they derive a cut set of all event sequences leading to the target state, i.e., the state representing the observation. These cut sets then constitute the diagnoses[2]. Our work differs to the approach by Kuntz et al. [Kun+11] in two ways: first, in this chapter we define an entire process for diagnosis in industrial applications and we only rely on the failure assessment itself without the need for additional documents revealing the internal structure of the system. Second, we generate a propositional Horn representation instead of an automaton to capture the faulty system behavior.

The remainder of this chapter is structured as follows: First, we propose a process that relies on failure assessments in order to develop diagnostic system models while keeping knowledge acquisition affordable. Second, we analyze two types of assessment, i.e, FMEA and FTA, that can be used to automatically infer abductive models. For each type of failure analysis a modeling methodology is introduced generating a system formalization suitable for abductive reasoning. Further, we give an overview of advantages and disadvantages of each assessment in regard to the resulting model and its capabilities within the entire troubleshooting task. Then, we discuss in more detail the fault identification portion of our abductive process. Section 3.3.1 presents initial empirical results for models generated on

[2]Note here, that since the cut sets are sets instead of sequences, the event ordering is lost.

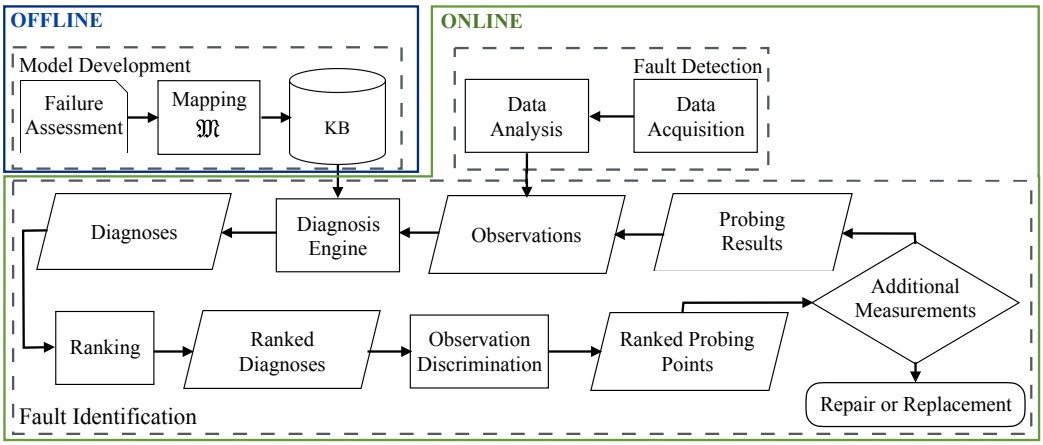


Figure 3.1: Process for incorporating abductive model-based diagnosis in an industrial setting [Koi+18].

top of FMEAs based on a practical and an artificial benchmark. Lastly, we discuss general limitations of our approach and conclude the chapter.

3.2 Defining a General Abductive Diagnosis Process

Intercalating abductive diagnosis into real-world applications faces several issues [TMM98; Mil+00]. We recognize the difficulties of integrating model-based diagnosis into existing work flows, constructing the domain model, and the complexity of abduction as the main obstacles. In this regard, we define a process addressing these two problems by dividing the problem into subtasks and separate modules, taking advantage of information available, and relying on the structure of the resulting system descriptions top compute solutions efficiently. We divide the approach into three main steps as can be seen in Figure 3.1:

1 **Model Development**

2 **Fault Detection**

3 **Fault Identification**

Here, we give a short overview of the stages and subsequently elaborate on certain parts in the following sections.

1 **Model Development.** As mentioned earlier abductive model-based diagnosis relies on an explicit description of the system behavior in presence of a fault. Our modeling methodology utilizes failure analyses available to compile the information recorded automatically into abductive models offline. As these assessments capture knowledge on failures and their symptoms, the mapping to a corresponding abductive *KB* is straightforward. Some additional considerations have to be made in order to create an appropriate model which we discuss throughout the chapter.

Since abductive diagnosis depends on the premise of model completeness, we assume that all significant fault modes for each contributing part of the system are being considered in the analysis[3]. Furthermore, our mapping approach expects consistent effect descriptions, i.e., a symptom is described in a uniform way throughout the assessment.

[3]Note here that this assumption is not particular to our approach, but is a prerequisite for model-based diagnosis in general [FB+97].

2 **Fault Detection.** Abduction derives possible explanations for observed anomalies. Consequently, to initiate the troubleshooting, the presence of a fault has to be detected. Within our process, we assume the manifestation of a fault is discovered by a failure detection method, e.g., a monitoring system, and therefore we do not consider data acquisition or analysis in detail.

3 **Fault Identification.** Once the presence of a disturbance has been established, the possible causes associated with the observations are to be computed. Due to the knowledge represented in the types of failure assessments we are considering, abductive diagnosis poses an intuitive approach for fault identification. We already discussed one possible algorithm capable of computing abductive diagnoses in the previous section. The process, however, is not limited to the use of this exact procedure [KW15a]. In the upcoming chapters, we will elaborate on other techniques to derive explanations.

In the course of this chapter, we explain further improvements to the initial set of solutions. We show a simple diagnoses ranking according to probability theory and how to determine the next probing point in order to diminish the number of solutions.

3.3 Integration of Failure Assessments into the Diagnostic Process

The initial construction of the system description related to model-based diagnosis constitutes a disadvantage. To automate this task, we propose a mapping function associating entries from failure assessments to propositional clauses to form an abductive *KB* [Wot14]. Risk analysis tools, such as FMEA, provide a systematic and comprehensive identification, review, and evaluation of possible threats or hazards. Usually, the assessment comprises the risk sources, consequences, magnitude, and likelihood [Car14]. Hence, they pose suitable information sources for abductive fault identification.

In this work, we focus the modeling process based on FMEA as well as FTA and show how these documents can be used for model generation. The upcoming subsections describe the methods to translate the knowledge available into a propositional Horn theory as well as the advantages and disadvantages of each failure assessment in regard to constituting the basis of an abductive knowledge base.

3.3.1 Failure Mode Effect Analysis

FMEA is an established standardized reliability tool in which components are systematically assessed in regard to their possible failure modes. These rigorous and comprehensive reliability and safety design evaluations are usually required by industry standards or government policies [Vog+14]. Generally, the objectives of an FMEA include the identification and prevention of failures and safety hazards, minimization of performance loss, development of preventive maintenance plans, and the usage for diagnostic techniques [Car14]. The reasoning is performed in an inductive bottom-up manner, where the general theories are derived from detailed examples on various levels of depth. Besides determining component-based single faults, each failure mode is examined in regard to its causes, detection mechanisms, and consequences. Further, the assessment encompasses fault probabilities and severity ratings. As detailed knowledge about the system structure, the requirements, the behavior of the components, and their operational relationship is necessary, such a review is conducted by a team of specialists [RH04].

In case the criticality of each failure mode is identified the technique is referred to as Failure Mode, Effects, and Criticality Analysis (FMECA). The criticality of a failure quantifies how hazardous or serious the caused interference is and is determined by the occurrence likelihood and the severity of the fault. Even though there are slight differences between the various FMEA/FMECA standards [Vog+14], both analyses result in a detailed document with a tabulation of components and their single point failures as well as consequences. Frequently, the results can be utilized qualitatively as a hazard analysis method or quantitatively considering the various ratings. The parts of an FMEA differ depending on the followed standard or guideline. However, certain parts are usually incorporated [Car14]:

- Component/Item: The component or item determines the artifact being analyzed.
- Failure Mode: The potential failure mode encompasses the way in which a component may neglect to deliver its intended function with the desired performance at various levels of depth within the system.
- Failure Cause: A failure cause describes the internal and external influences as well as their interaction which may lead up to the failure mode such as wear, aging, defective material, or damage.
- Failure Effect: Effects are failure consequences and have to be considered at various levels such as local effects or impacts on the overall system and operation.
- Detection Method and Rating: Often a particular detection mechanism is recorded, such as automated warnings as well as alarms or the discovery by a human operator. Additionally, a rating is applied to account for the detectability of a failure, i.e., the likelihood of discovery. Evident failures are detected instantly when they occur, while hidden failures can usually only be confirmed through testing.
- Severity: To quantify the analysis a severity rating is included, based on the most serious failure consequences.
- Occurrence: The occurrence rating associates a failure with the likelihood of its presence.
- Risk Priority Number (RPN): The RPN is the arithmetic product of severity, occurrence and detection ratings.
- Actions: The actions are recommended efforts to reduce or eliminate the risk associated with the cause of a failure.

Example 3.1

Table 3.1 depicts an excerpt of an exemplary FMEA for the yaw drive of a wind turbine. As can be seen, not all columns described previously are present within the analysis. Yet, as we will see the most important information for the mapping, i.e., component, failure mode and effects, isre included.

Table 3.1: FMEA excerpt (adapted from [Rad+93]).

Component	Failure Mode	Failure Effect	Failure Cause	Likelihood	Severity	Detection Method

Yaw Drive	Fails to rotate	No yaw, failure of safety system, decrease of efficiency	Motor not electrically powered, Motor burned after emergency stop, Yaw drive disconnected from frame, mechanical damage, fatigue fracture, capacitors burned during emergency stop	2.2E-5	3	Visual inspection
Yaw Drive	Yaw shaft blocked	No yaw, decrease of efficiency	Motor not electrically powered, Motor burned after emergency stop, Yaw drive disconnected from frame, mechanical damage, fatigue fracture, capacitors burned during emergency stop	1.3E-5	3	Visual inspection

Model Development

Formally, we create an abductive knowledge base. As the failures recorded in the FMEA represent the connections between single faults and a conjunction of effects, we can create a theory consisting of definite propositional Horn clauses [Wot14]. This limitation of the expressiveness of the logical model leads to an avoidance of some of the computational inefficiencies inherent to abduction in general. For model creation, we simplify the FMEA to three essential columns, namely the one featuring the set of components *COMP*, their potential failure modes *MODES*, and the set of failure effects forming a subset of the set of propositional variables *PROPS*.

Definition 3.1 (Failure Mode Effect Analysis (FMEA) [Wot14]). An *FMEA* is a set of tuples $(\mathbb{C}, \mathbb{M}, \mathbb{E})$ where $\mathbb{C} \in$ *COMP* is a component, $\mathbb{M} \in$ *MODES* is a failure mode, and $\mathbb{E} \subseteq$ *PROPS* is a set of effects.

As abductive reasoning relies on a formalization of failures and their symptoms, the conversion of an FMEA to a propositional *KB(A,Hyp,Th)* is straightforward. First, we create the set of hypotheses. In the FMEA each component-failure mode pair represents a possible cause. Each pair is mapped to a propositional variable $mode(\mathbb{C}, \mathbb{M})$, where $\mathbb{C}$ is the component and $\mathbb{M}$ is the failure mode. This propositional variable is then added to the set *Hyp*. Equation (3.1) defines *Hyp* in this modeling context.

$$Hyp =_{def} \bigcup_{(\mathbb{C},\mathbb{M},\mathbb{E}) \in FMEA} \{mode(\mathbb{C}, \mathbb{M})\} \tag{3.1}$$

Second, the set A then is simply the union over all hypotheses as well as propositional variables representing effects as shown in Equation (3.2).

$$A =_{def} \bigcup_{(\mathbb{C},\mathbb{M},\mathbb{E})\in FMEA} \mathbb{E} \cup \{mode(\mathbb{C},\mathbb{M})\} \tag{3.2}$$

Example 3.1 (cont.)

Considering the FMEA in Table 3.1, there are two component-failure mode pairs forming the set *Hyp*:

$$Hyp = \left\{ \begin{array}{c} mode(Yaw\_Drive,\ Fails\_to\_rotate), \\ mode(Yaw\_Drive,\ Shaft\_blocked) \end{array} \right\}$$

The set of all propositional variables then contains the hypotheses as well as all propositional variables constituting effects.

$$A = \left\{ \begin{array}{c} mode(Yaw\_Drive,\ Fails\_to\_rotate), \\ mode(Yaw\_Drive,\ Shaft\_blocked), \\ no\_yaw,\ failure\_safety\_system, \\ decrease\_of\_efficiency \end{array} \right\}$$

Lastly, the propositional theory is determined by the relation between defects and their manifestations as depicted in the FMEA. For each record of the FMEA the mapping function $\mathfrak{M}$: $2^{FMEA} \mapsto HC$ generates a Horn clause as a subset of the set of Horn clauses HC.

Definition 3.2 (Mapping function $\mathfrak{M}$ [Wot14]). Given an FMEA, the function $\mathfrak{M}$ is defined as follows:

$$\mathfrak{M}(FMEA) =_{def} \bigcup_{t\in FMEA} \mathfrak{M}(t) \tag{3.3}$$

where

$$\mathfrak{M}(\mathbb{C},\mathbb{M},\mathbb{E}) =_{def} \{mode(\mathbb{C},\mathbb{M}) \rightarrow e \,|\, e \in \mathbb{E}\} \tag{3.4}$$

Example 3.1 (cont.)

The theory then simply comprises Horn clauses where a single hypothesis implies one of its effects.

$$Th = \left\{ \begin{array}{c} mode(Yaw\_Drive,\ Fails\_to\_rotate) \rightarrow no\_yaw, \\ mode(Yaw\_Drive,\ Fails\_to\_rotate) \rightarrow failure\_safety\_system, \\ mode(Yaw\_Drive,\ Fails\_to\_rotate) \rightarrow decrease\_of\_efficiency, \\ mode(Yaw\_Drive,\ Shaft\_blocked) \rightarrow no\_yaw, \\ mode(Yaw\_Drive,\ Shaft\_blocked) \rightarrow decrease\_of\_efficiency \end{array} \right\}$$

One Single Fault Diagnosis Property

The appropriateness of the models obtained from the FMEA is yet to be examined. Due to the fact that abductive explanations are consistent by definition and complete given an exhaustive search, suitability refers to the characteristic of the model that given all necessary information a single diagnosis can be computed. We refer to this feature as the One Single Fault Diagnosis Property (OSFDP).

Definition 3.3 (One Single Fault Diagnosis Property (OSFDP) [Wot14]). Given a *KB* (*A*, *Hyp*, *Th*). *KB* fulfills the OSFDP if the following hold: $\forall m \in Hyp : \exists Obs \subseteq A : \{m\}$ is a diagnosis of (*A*,*Hyp*,*Th*,*Obs*) and $\neg\exists m' \in Hyp : m' \neq m$ such that $\{m'\}$ is a diagnosis for the same *PHCAP*.

The property can be checked by computing for each $h \in Hyp$ the set of propositions $\delta(h)$, such that $\{h\} \cup Th \models \delta(h)$. In case $\{h\} \cup Th$ leads to a contradiction, $\delta(h)$ equals $\emptyset$. If for any two hypotheses the derived propositions are the same, the OSFDP is not satisfiable. Besides determining whether single fault diagnoses can be computed, the absence of the property indicates that the *KB* is not complete, i.e., information is missing. In the case of FMEAs this can signal that internal variables or observations have not been contemplated during the analysis. Wotawa [Wot14] provides the polynomial time algorithm CHECKOSFDP for testing whether the property is satisfied.

Algorithm 3.1: CHECKOSFDP [Wot14]

Require: *A*: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: the Horn theory
Ensure: *true* if the *KB* satisfies the OSFDP property, and *false* otherwise

1: **for all** $h \in Hyp$ **do**
2: Let $\delta(h) \subseteq A$ be the largest set such that $\{h\} \cup Th \models \delta(h)$, if $\{h\} \cup Th \models \perp$ let $\delta(h) = \emptyset$.
3: **end for**
4: **for all** $h_1 \in Hyp \wedge \delta(h_1) \neq \emptyset$ **do**
5: **for all** $h_2 \in Hyp \wedge h_1 \neq h_2 \wedge \delta(h_2) \neq \emptyset$ **do**
6: **if** $\delta(h_1) = \delta(h_2)$ **then**
7: **return** *false*
8: **end if**
9: **end for**
10: **end for**
11: **return** *true*

Example 3.1 (cont.)

For our running example, CHECKOSFDP returns *true*, since δ(*mode(Yaw_Drive,Fails_to_rotate)*) $= \{$*no_yaw*,*failure_safety_system*, *decrease_of_efficiency*$\}$ and δ(*mode(Yaw_Drive, Shaft_blocked)*) $= \{$*no_yaw*, *decrease_of_efficiency*$\}$. Hence, the two fault modes can be distinguished by the intersection of their δ sets, i.e., based on whether there is a failure in the safety system or not, we can determine which fault has occurred.

A simple procedure to enforce the OSFDP treats indistinguishable faults as a unit. Hence, each set of indistinguishable hypotheses $\{h_1, h_2, \ldots h_n\}$ is replaced by a new hypothesis h'. We proceed with these substitutions until the OSFDP is fulfilled. Algorithm DISTINGUISHHYPOTHESES ensures that after termination the given *KB* satisfies the property. It assumes that for each hypothesis in *Hyp* the set $\delta(h)$ has already been computed. Due to the finite number of hypotheses as well as possible effects contained in $\delta(h)$, the procedure must halt. Further, the complexity of the algorithm is determined by the three nested loops, hence $O(|Hyp|^2 + |A - Hyp|)$.

Algorithm 3.2: DISTINGUISHHYPOTHESES

Require: A: the set of propositional variables, Hyp: the set of hypotheses, Th: the Horn theory
Ensure: $KB(A,\Psi,Th)$: knowledge base satisfying the OSFDP property

1: $\Psi[|Hyp|] \leftarrow Hyp$
2: **for all** $h_1 \in \Psi$ **do**
3: **for all** $h_2 \in \Psi$ **do**
4: **if** $h_1 \neq h_2$ **then**
5: **if** $\delta(h_1) = \delta(h_2)$ **and** $\delta(h_1) \neq \emptyset$ **then**
6: Create new hypothesis h' ▷ $h' \notin Hyp$
7: $\Psi \leftarrow h' \cup \Psi$
8: $A \leftarrow h' \cup A$
9: **for all** $e \in \delta(h_1)$ **do**
10: $Th \leftarrow (h' \rightarrow e) \cup Th$
11: $Th \leftarrow Th \setminus (h_1 \rightarrow e)$
12: $Th \leftarrow Th \setminus (h_2 \rightarrow e)$
13: **end for**
14: $\Psi \leftarrow \Psi \setminus \{h_1\} \setminus \{h_2\}$
15: $A \leftarrow A \setminus \{h_1\} \setminus \{h_2\}$
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **return** $KB(A,\Psi,Th)$

Enforcing the OSFDP has a practical rational: Since the indistinguishable faults cannot be differentiated, all components have to be repaired or replaced in case they are part of the diagnosis. Thus, treating them as a single unit during diagnosis does not influence the result; however, it does have an effect on the computational effort because it reduces the number of possible hypotheses to consider.

Complexity

Due to the topology of FMEAs the models constructed via $\mathfrak{M}$ exhibit a specific composition. It is apparent that the corresponding models are acyclic and further feature a forward structure from causes to effects. In particular, the hypotheses and symptoms are disjoint sets and implications connect one hypothesis to one effect, thus are bijunctive definite Horn clauses[4] with one exception; in case we want to state that an effect and its negation cannot occur at the same time we introduce $e \wedge n\_e \rightarrow \bot$ where e is an effect and n_e represents its negation. When using an ATMS to derive diagnoses there is another exception; the implication that maps all effects to be observed to a special proposition *ex* (for details see Section 2.3.1).

In regard to the computational complexity of computing the abductive solutions relating to the generated models let us assume a *PHCAP* with $h_1, \ldots, h_n$ hypotheses potentially connected with $e_1, \ldots, e_m$ effects. Some of the effects are observable and thus lead to the proposition *ex*. Figure 3.2 illustrates such a *PHCAP* as an And-or-graph. When using an ATMS for implementing a solver, we have a similar representation, where the context

[4]That is each clause contains, exactly one positive and one negative literal, i.e., each clause represents an implication from a hypothesis to a propositional variable from $A \setminus Hyp$.

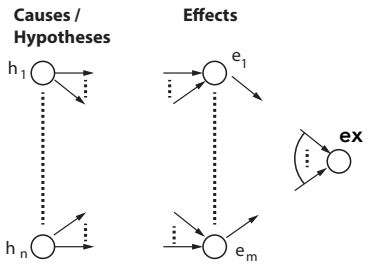


Figure 3.2: And-or-graph representation of the *PHCAP* comprising hypotheses $\{h_1, \ldots, h_n\}$ and effects $\{e_1, \ldots, e_m\}$ where some of the latter can be observed.

information is stored in the labels of the graph's vertices. The context is the set of hypotheses or assumptions that have to be true in order for the vertex to become true. In ATMS terminology this would refer to an environment. Each hypothesis h_i has a single context $\{\{h_i\}\}$, whereas an effect might have more. Due to the special structure of the model *Th*, each environment of an effect can only comprise exactly one hypothesis. In the worst case an effect e_j has the label $\{\{h_1\}, \ldots, \{h_n\}\}$. Computing the label of *ex* incrementally using an ATMS, results in parametrized worst case computational complexity of $O(n^m)$. When combining two effects, n times n contexts have to be build. Incorporating another effect leads to n^3 contexts and so on. Hence, when not considering negated effects, the computation of all solutions can be done in polynomial time. While m can become quite large in theory the empirical results in the subsequent section suggest that in practice the computational effort is limited. Implications of the form $e \wedge n\_e \rightarrow \bot$ do not increase the computational complexity. We first compute all labels disregarding these implications and afterwards remove all contexts, which have thus become inconsistent. Again this requires polynomial time. The following theorem summarizes these findings.

Theorem 3.1. Given an FMEA. When applying $\mathfrak{M}$ the corresponding *PHCAP* can be solved in polynomial time $O(n^m)$ where n is the number of hypotheses ($n = |Hyp|$) and m the maximum number of effects to be observed.

This result complies with the computational complexity of definite Horn abduction problems shown in Nordh and Zanuttini [NZ08] with the addition of handling negated effects. It is worth noting that the runtime complexity in the Horn clause abduction case would be $O(2^n)$ for finding all parsimonious solutions. A consequence of the theorem is that the stated solution is tractable provided that m is small.

Advantages and Limitations

The application areas of FMEA are widespread as it has been applied to complex systems. Its representation can form a comprehensive knowledge base for failures in each part of the system, explaining their effects on subcomponents that are dependent on them and the entire system [HW98]. Thus, the mapping to an abductive model can be automated in a simple manner. Due to the structure of the FMEA, the resulting system descriptions are acyclic and contain solely bijunctive definite Horn clauses, i.e, each clause is an implication leading from one hypothesis to a single effect variable. These types of models require polynomial time when used for computing abductive diagnoses. In addition, as FMEA usually considers single faults, the resulting diagnostic system holds the single fault assumption. Furthermore, the analysis contains additional information, such as data on failure occurrence likelihood or severity, that can be incorporated in the diagnosis process for prioritizing probable or severe defects. However, these ratings are often subjective, thus they should merely be

considered as a means to focus on a subset of diagnoses and not as a discrimination criteria. In Section 3.4, we discuss strategies to improve the initial diagnosis result, where we can utilize this auxiliary knowledge stored in an FMEA.

Since abductive diagnosis depends on the premise of model completeness, we assume that all significant fault modes for each contributing part of the system are being considered in the analysis. Moreover, our mapping approach expects consistent effect descriptions, i.e., a symptom is represented in a uniform way throughout the FMEA. Of course, in order to count as an observation within the diagnostic process, each effect mentioned in the analysis has to be detectable in nature. FMEA does not take into account the potential interdependencies between various failure modes and effects. While the absence of interconnections between failures may apply to some systems or subsystems, a generalization is not correct. Thus, depending on the underlying assessment artifact, the analysis might not depict the causal relations in its entirety. Furthermore, as the set of effects corresponding to a failure is represented by a conjunction, a contrary measure of any of these manifestations results in a discrimination of the failure mode. This observation is essential in the context of sensor errors and noise, since those conditions can provoke an incorrect rejection of a diagnosis.

Example 3.1 (cont.)

Assume for Example 3.1 that even though there is no yaw and a decrease in efficiency, the safety system does not fail. Then the Δ-Set would contain *mode(Yaw_Drive, Shaft_blocked)* but not *mode(Yaw_Drive, Fails_to_rotate)* as the Horn theory requires in case of the yaw drive failing to rotate that the safety systems fails.

Initial Empirical Evaluations

In order to determine the feasibility of the modeling approach and Algorithm ABDUCTIVE-EXPLANATIONS, we carried out two experiments. Both evaluations were performed on an Intel Core i7-4700MQ processor (2.60 GHz) with 8 GB RAM on Windows 7 Enterprise (64-bit) exploiting a Java implementation of the mapping function $\mathfrak{M}$ and a Java implementation of the unfocused ATMS [FK88] to derive the explanations.

In the first experiment, we tackled two benchmarks; on the one hand we used available FMEAs to obtain abductive diagnosis problems and on the other hand created artificial examples in order to reach practical computational limitations. For the real-world examples, we used eleven different FMEAs and created their corresponding abductive knowledge base *KB* through the mapping function $\mathfrak{M}$. We obtained several publicly available as well as project internal FMEAs covering various technical systems and subsystems such as electrical circuits, a connector system by Ford, Focal Plane Unit of the Heterodyne Instrument for the Far Infrared built for the Herschel Space Observatory (HIFI-FPU), the Anticoincidence Detector mounted on the Large Area Telescope of the Fermi Gamma-ray Space Telescope (ACD), printed circuit boards (PCB), the the Maritim ITStandard (MiTS), as well as rectifier, inverter, transformer, and backup components of an industrial wind turbine. These FMEAs vary in the number of hypotheses, effects, and rules as can be seen in Table 3.2. In order to create a *PHCAP* and compute abductive explanations we require a set of observations, thus we randomly selected between 1 to $|A \setminus Hyp|$. For our diagnosis purposes, we further generated additional rules and propositions as described in previous sections, e.g. $e \wedge n\_e \rightarrow \bot$. Table 3.2 provides the combined results from several test runs. Note that the numbers referred in the table corresponds to the number in the underlying FMEAs and not to the abductive model.

Table 3.2: Features of the failure mode and effect analyses and experiment results of the first experiment.

	FMEA Structure			Runtime [in ms]				#Diagnoses	
	#Hypotheses	#Effects	#Rules	MIN	MAX	AVG	MED	MAX	AVG
ACD	13	16	52	<1	4	<1	<1	11	2.60
Electrical circuit	32	17	52	<1	379	30.70	1	792	191.86
Ford connector system	17	17	56	<1	2	<1	<1	42	3.35
HIFI-FPU	17	11	35	<1	14	<1	<1	63	7.72
MiTS 1	17	21	47	<1	1	<1	<1	12	5.47
MiTS 2	22	15	48	<1	64	1.70	<1	288	35.52
PCB	10	11	24	<1	1	<1	<1	2	1.62
Inverter	29	38	165	<1	472	8.13	5.2	500	32.52
Rectifier	20	17	93	<1	40	1.61	<1	96	15.84
Transformer	4	8	22	<1	1	<1	<1	4	1.175
Backup components	25	30	114	<1	93	3.34	2.7	264	26.11

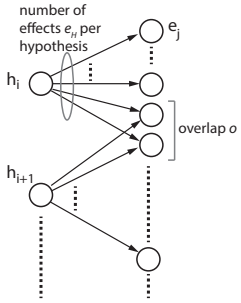


Figure 3.3: Structure of the artificial examples used for the first evaluation.

The second portion of the experiment focuses on artificial examples. Figure 3.3 depicts their principle structure. We assume n hypotheses, each connected with e effects. The degree of effects overlapping from one hypothesis h_i and its neighbor hypothesis h_{i+1} is denoted by the parameter o. The parameter k which is not illustrated in Figure 3.3 refers to the number of effects that have to be explained. In our representation, this is the number of arcs leading from an effect to a node representing the proposition *ex*. For the synthetic example category, we assume that whenever possible such an arc should be from effects that correspond to different hypotheses.

For our practical examples, we reported maximum runtimes from less than a second, as revealed in Table 3.2. Generally, as expected, it can be observed that the runtime correlates to the number of hypotheses considered and hence subsequently the possible diagnoses size. In regard to the artificial examples, we were first interested in obtaining the runtime as a function of the number of rules, where we vary all other parameters. Figure 3.4 shows the obtained results that are based on 5,749 generated examples. As can be seen, there is a large variation of the runtime. Though, in most cases even when handling $20,000$ rules the runtime did not exceed one second, there were examples causing an extensive computation time, which can be found in the upper left part of Figure 3.4. In order to explain this behavior we had a look at the number of environments of *ex*. Figure 3.5 depicts the runtime as a function of the number of environments computed for *ex* and shows that the runtime and the number of environments highly correlate with each other. For our category of examples, the number of computed environments depends on the overlap o and the parameter k.

In the second experiment, we aimed at showing the advantage of the OSFDP property. Hence, we utilized the same practical benchmark as exploited the first experiment with an additional analysis concerning the main bearing of a wind turbine. We tested each FMEA for the OSFDP and as Table 3.3 reveals none, except of the model resulting from the

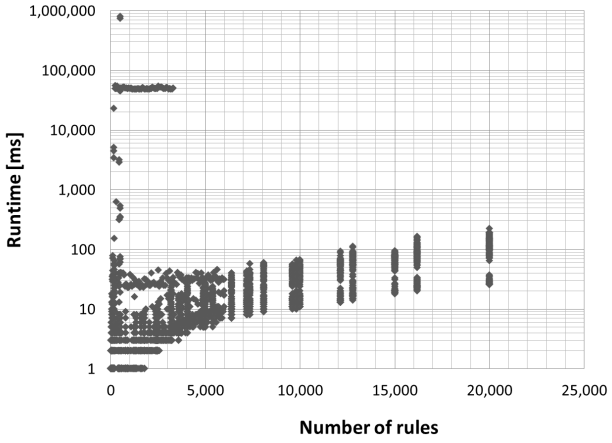


Figure 3.4: Runtime versus number of rules of the first experiment.

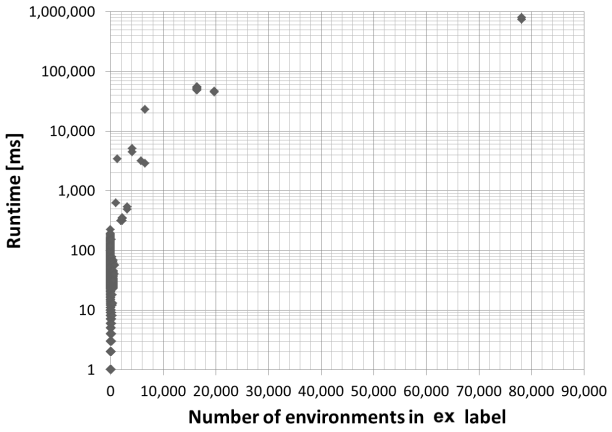


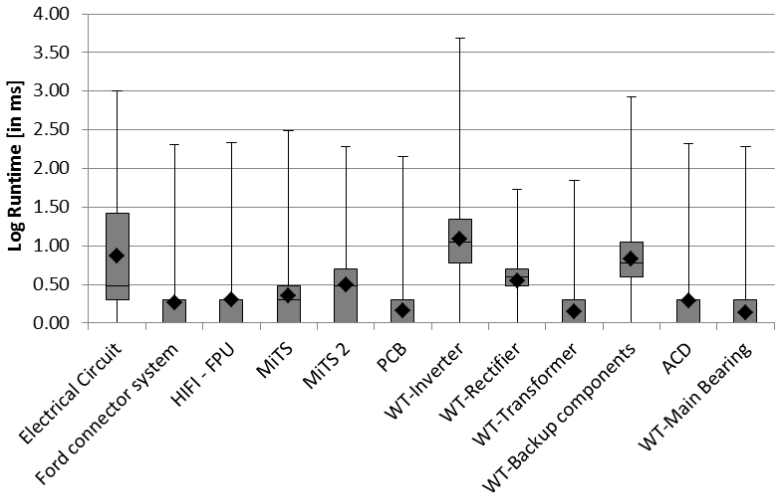
Figure 3.5: Runtime versus number environments in the observation node of the first experiment.

transformer's failure assessment, of the original models satisfies the property. To generate models which fulfill the OSFDP, each set of indistinguishable hypotheses was exchanged with a new single hypothesis representing said set. Thus, the number of hypotheses and rules diminishes for the adapted models. We do not report on the computation time of the mapping, as model generation is executed offline and the conversions we have computed so far took less than a second.

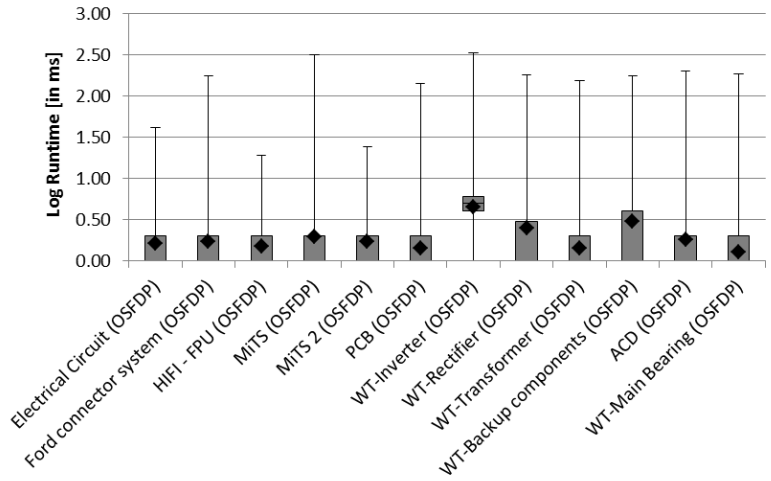
After model compilation, we examined the performance of a Java implementation of Algorithm ABDUCTIVEEXPLANATIONS on the generated *KB*s. For each FMEA we ran the algorithm for $|Obs|$ from 1 to maximum number of effects possible. The observation set was generated randomly, however, we utilized the same observations for the original as well as for the adapted model. The results reported in Table 3.3 have been obtained from 100 trials. Unsurprisingly, the runtime increases with the number of rules to consider. As the results show while there are maximum computation times of around five seconds, the median of the distributions is located around and below ten milliseconds. Comparing the original model to the OSFDP variant, we see a performance advantage of the latter for the majority of examples. It is worth noticing that even though the transformer example already satisfied the OSFDP, the runtimes deviate. These discrepancies can be attributed to the small unit of measurement in the millisecond range. Figure 3.6 depicts the underlying statistical distribution of the performance for the original and the adapted models. In order to determine whether the

Table 3.3: Features of the FMEAs and empirical results for the second experiment. For each component we conducted the evaluation using the original model as well as a model fulfilling the OSFDP. The last three columns display the maximum number of single faults, double faults, and triple faults, respectively.

Component		Model Structure			Runtime [in ms]				#Diagnoses				
		#Hyp	#Effects	#Rules	MIN	MAX	AVG	MED	MAX	AVG	SF	DF	TF
Electrical circuit	Original	32	17	52	< 1	994	48.04	2	792	191.61	11	22	44
	OSFDP	15	17	35	< 1	40	0.99	1	1	1.00	1	1	1
Ford connector system	Original	17	17	56	< 1	204	2.08	1	18	3.14	6	18	18
	OSFDP	15	17	49	< 1	172	1.37	1	18	2.85	6	12	18
HIFI-FPU	Original	17	11	35	< 1	214	5.17	1	189	8.21	7	21	27
	OSFDP	9	11	27	< 1	18	0.83	1	12	1.59	3	6	6
MiTS 1	Original	17	21	47	< 1	307	7.59	1	12	5.40	3	3	6
	OSFDP	13	21	43	< 1	312	5.38	1	1	1.00	1	1	1
MiTS 2	Original	22	15	48	< 1	191	6.60	2	288	37.44	8	24	24
	OSFDP	14	15	37	< 1	23	1.04	1	10	1.96	5	10	10
PCB	Original	10	11	24	< 1	140	1.29	0	2	1.55	2	2	2
	OSFDP	9	11	23	< 1	140	0.87	0	1	1.00	1	1	1
ACD	Original	13	16	52	< 1	210	4.47	1	15	2.44	5	8	15
	OSFDP	12	16	39	< 1	199	3.51	1	10	1.77	5	5	10
Inverter	Original	29	38	165	< 1	4,830	34.80	10	1,280	33.00	19	57	76
	OSFDP	23	38	124	< 1	331	9.91	4	144	6.04	13	39	26
Rectifier	Original	20	17	93	< 1	53	3.80	3	160	10.76	15	40	64
	OSFDP	14	17	66	< 1	176	4.11	2	30	3.50	9	18	24
Transformer	Original	4	8	22	< 1	70	0.73	0	4	1.17	4	2	2
	OSFDP	4	8	22	< 1	153	1.05	0	4	1.17	4	2	2
Backup components	Original	25	30	114	< 1	856	14.77	5	864	25.08	9	42	210
	OSFDP	19	30	95	< 1	172	3.67	3	90	3.53	7	30	72
Main bearing	Original	3	5	20	< 1	191	1.68	0	3	2.41	3	0	0
	OSFDP	2	5	15	< 1	184	0.84	0	2	1.41	2	0	0



(a) Original models.



(b) Adapted models fulfilling the OSFDP.

Figure 3.6: Box-and-whisker plots of the underlying statistical distributions of the log runtimes for the second experiment.

adapted models are superior in regard to the diagnosis performance, we used an adaptation of the sign test as described by Stumptner and Wotawa [SW01]. Suppose paired runtime data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ from the original and adapted models, respectively. We propose the hypothesis $H_0 : mX = mY$, stating a median difference of zero. $H_1 : m_d > 0$ is our alternative hypothesis, where m_d denotes the median of $X - Y$. Let Z be the sum of pairs, where $x_i > y_i$. Given H_0 is true, the test statistic $Z \sim B_{0.5,n}$ has to be binomial distributed. We refute H_0 and accept H_1 if the critical value z_α is smaller than the z value from the sample. Since there is a large number of samples in our evaluation, the critical values for the sign test are not based directly on the binomial distribution, but rather on a normal approximation. For $\alpha = 0.05$ we accepted H_1, i.e., the runtime performance for the adapted models is superior to the original ones.

3.3.2 Fault Tree Analysis

Fault trees provide a systematic sequence of events leading to an incident of interest, i.e., the top event. By employing a top-down approach the analysis reasons from effects to causes. Starting at the root the chain of events prompting the undesired event at the top is determined in a deductive manner. Besides basic events forming the leafs of the fault tree, intermediate incidents leading up to the top event are considered [RH04]. Logic gates describe the relations between these events. Each gate has a set of basic or intermediate failures as input and the output is defined by a single event. Thus, the tree represents logical paths of cause-effect relationships [RS15].

Based on information on event likelihood, the tree can be quantified. To compute the frequency of the top event, the probabilities of each output event are determined by the gates and the input events' probabilities in a bottom-up manner. Thus, in case a quantification is desired, the probabilities of the basic events have to be know [RH04].

Example 3.2

Figure 3.7 shows an exemplary systematic chain of events leading to insufficient lubrication of a gearbox in a wind turbine. The depicted fault tree has a top event (*Insufficient lubrication*), three basic events (*Oil filter failure*, *Leakage/Cracks of cooler* and *Rusty cooling fins of radiator*), and one intermediate event (*Poor cooling*). These incidents are connected by an OR as well as an AND gate. The former states that either the *Leakage/Cracks of cooler* or *Rusty cooling fins of radiator* or both must occur to cause *Poor cooling*, while the latter requires both *Poor cooling* and *Oil filter failure* for *Insufficient lubrication* to arise.

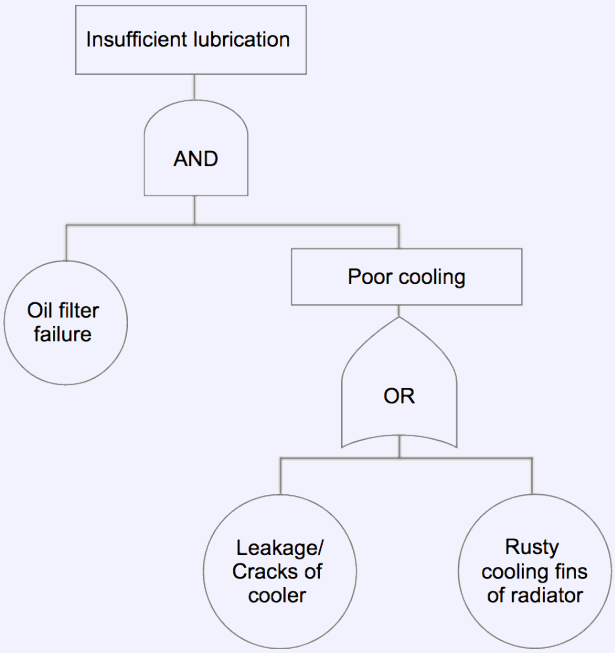


Figure 3.7: Fault Tree (adapted from [Már+12; Bot+12]).

Model Development

While there are several logic gates and symbols available in fault trees, we focus in our analysis on a simple fault tree representation with three types of events (basic, intermediate,

top) and the two most common gates, i.e., AND and OR. We assume that each system is described by a set of fault trees $\mathbb{T}$, where each fault tree describes the occurrence combinations leading to a top event that represents a certain effect of interest.

Definition 3.4 (Fault Tree (FT)). A fault tree *FT* is a pair $(G, \mathcal{E})$ where G is a set of logic gates and $\mathcal{E}$ a set of events. $\mathcal{BE} \subset \mathcal{E}$ is the set of basic events, while $\varepsilon \in \mathcal{E}$ is the top event. $\Omega(I,\omega) \subseteq G$ denotes the set of OR gates and $\mathcal{A}(I,\omega) \subseteq G$ the set of AND gates, where $I \subseteq \{\mathcal{E} \setminus \varepsilon\}$ is the set of input events and $\omega \in \{\mathcal{E} \setminus \mathcal{BE}\}$ is the output event.

As a fault tree is a pictorial representation of a Boolean formula depicting how the top event is caused by other events, we can simply transform the tree into a set of clauses by determining the Boolean expression for each gate. Each event corresponds to a propositional variable in *A*. In our case, we assume that the basic events represent the initial root causes, thus, we limit the set of hypotheses to only comprise propositional variables corresponding to these primary incidents. The relevant sets *A* and *Hyp* are defined the following way:

$$A =_{def} \bigcup_{e \in \mathcal{E}} \{e\} \tag{3.5}$$

$$Hyp =_{def} \bigcup_{\beta \in \mathcal{BE}} \{\beta\} \tag{3.6}$$

In case intermediate events depict failures that should be diagnosable, we would define the set *Hyp* as $\{\mathcal{E} \setminus \epsilon\}$.

We define a mapping function $\mathfrak{M}_{FTA} : 2^{FT} \mapsto HC$, creating for each fault tree a set of Horn clauses based on the gates comprising the tree.

$$\mathfrak{M}_{FTA}(FT) =_{def} \bigcup_{g \in G} \mathfrak{M}(g) \tag{3.7}$$

where

$$\mathfrak{M}_{FTA}(g) = \begin{cases} \bigcup_{i \in I} \{i \rightarrow \omega\}, & g \in \Omega(I,\omega) \\ \bigwedge_{i \in I} i \rightarrow \omega, & g \in \mathcal{A}(I,\omega) \end{cases} \tag{3.8}$$

In case of an OR gate, for each input event a Horn clause is created where the input implies the gate's output. As the AND gate represents the relation that all inputs have to be present in order for the output event to occur, a single implication is added, such that a conjunction of the variables representing the inputs leads to the output event. The abductive theory then is the union over all Horn clauses generated over the gates of the fault tree.

Example 3.2 (cont.)

For the fault tree in Figure 3.7, we can record the following set of propositional variables and hypotheses:

$$A = \left\{ \begin{array}{c} Insufficient\_lubrication, Oil\_filter\_failure, Poor\_cooling, \\ Leakage\_cracks\_of\_cooler, Rusty\_cooling\_fins\_of\_radiator \end{array} \right\}$$

$$Hyp = \left\{ Oil\_filter\_failure, Leakage\_cracks\_of\_cooler, Rusty\_cooling\_fins\_of\_radiator \right\}$$

Thus, considering the gates the resulting theory of the fault tree is as follows:

$$Th = \left\{ \begin{array}{c} Oil\_filter\_failure \wedge Poor\_cooling \rightarrow Insufficient\_lubrication, \\ Leakage\_cracks\_of\_cooler \rightarrow Poor\_cooling, \\ Rusty\_cooling\_fins\_of\_radiator \rightarrow Poor\_cooling \end{array} \right\}$$

Note that the resulting *KB*, however, encompasses solely the information of a single tree. Thus, to create an entire system model the knowledge bases resulting from each tree in the system have to be combined into a single *KB*.

Example 3.2 (cont.)

Computing the diagnosis of *Insufficient_lubrication*, we obtain

$$\Delta\text{-Set} = \left\{ \begin{array}{c} \{Oil\_filter\_failure, Leakage\_cracks\_of\_cooler\}, \\ \{Oil\_filter\_failure, Rusty\_cooling\_fins\_of\_radiator\} \end{array} \right\} .$$

Advantages and Limitations

Fault trees provide a clear and logical representation of cause-effect relations between combinations of events. In order to produce a meaningful abductive theory that can be utilized in diagnosis, each basic event has to represent a failure or cause. Further, for each diagnosable effect there has to be a fault tree where the top event represents the symptom. As with the FMEA a coherent description of the failures throughout the fault trees of the systems has to be guaranteed for an automatic model creation. Additionally, since it is a top-down approach, the analysis has to ensure that all contributors, i.e., failures, for a particular effect have been considered within the assessment to secure completeness.

Since we assume that the fault trees contain only AND as well as OR gates, the created theories can feature a disjunction and conjunction of hypotheses, thus, are Horn. In contrast to the FMEA models, however, they do not feature a bijunctive form. It is apparent that due to the knowledge encompassed within the fault tree, the resulting models are more expressive than the ones generated on top of an FMEA, i.e., the information stored in a fault tree can depict more relations than an FMEA due to the different logical connections that can be represented. In case of a quantitative analysis of the fault tree, the prior probability of the basic events are to be known. Thus, this knowledge can be utilized in the diagnostic context to compute diagnosis probabilities as we describe in Section 3.4.

The relation between FTA and abductive reasoning is worth mentioning. A (subset) minimal cut set encompasses a possible combination of basic events that lead to the top event [RH04]. As a minimal cut set is a prime implicant of the top event, it is equivalent to an abductive diagnosis for the observation of the top event. For instance, Rauzy and Dutuit [RD97] perform qualitative and quantitative FTA by computing minimal cut sets of operating on a Binary Decision Diagram (BDD). Another common strategy is to represent each gate as a Boolean expression of basic events or gates and converting it into DNF such that each conjunction represents a minimal cut set [RS15].

Example 3.2 (cont.)

Given the fault tree, we can obtain the Boolean formula ϕ represented by the tree, i.e.,

$$\phi = Oil\_filter\_failure \wedge (Leakage\_cracks\_of\_cooler \vee Rusty\_cooling\_fins\_of\_radiator) .$$

Converting this expression to DNF, we receive

$$\begin{aligned} \phi_{\text{DNF}} = \quad & (Oil\_filter\_failure \wedge Leakage\_cracks\_of\_cooler) \vee \\ & (Oil\_filter\_failure \wedge Rusty\_cooling\_fins\_of\_radiator) . \end{aligned}$$

Hence, the minimal cut sets are {*Oil_filter_failure,Leakage_cracks_of_cooler*} and {*Oil_filter_failure, Rusty_cooling_fins_of_radiator*}, which in fact are the diagnoses given the *KB(A,Hyp,Th)* constructed from the fault tree and $Obs = \{Insufficient\_lubrication\}$.

3.4 Fault Identification

Since the modeling methodology generates (bijunctive definite) Horn theories abductive reasoning is—depending on the underlying failure assessment—either tractable or not [NZ08]. Due to the structure of the FMEAs, the resulting system descriptions are acyclic and contain solely bijunctive definite Horn clauses. There are two exceptions: the formulae generated to account for contradicting observations and the implication mapping all observed effects to the proposition *ex* in the context of the ATMS. Further the intersection of the set of hypotheses and effects is empty. These features of the model all reduce the computation complexity in regard to the abduction problem. In particular, abductive diagnosis requires polynomial time in this case as shown in the previous section. For the models constructed on basis of FTA the complexity of propositional Horn abduction applies. In the preliminaries, we have already covered abductive reasoning with the ATMS and for now this mechanization to compute diagnoses should suffice. Other approach will be covered in the upcoming chapters. Subsequently, we focus on possibilities to improve the solutions returned by the diagnosis engine, i.e., observation discrimination and fault ranking.

3.4.1 Observation Discrimination

Generally, there might be an exponentially many diagnoses dependent on number of abducibles [Byl+91]. In a real world context, however, a single solution is preferred. Probe selection has been proposed as a way to minimize the number of results. The standard therapeutic approach presented by Friedrich, Gottlob, and Nejdl [Fri+90a] first computes a diagnosis Δ for the *PHCAP* and then checks each hypothesis of Δ whether it applies or not. Depending on the result of the validation, the *PHCAP* is modified. This approach does not compute all diagnoses before considering treatment, but is an interleaved process between repair and diagnosis. Checking whether a hypothesis applies would imply to check whether a component is faulty or not, hence replacing it. However, due to the component costs and the need for specialized equipment this is not a feasible approach in many practical application scenarios.

The method used by Wotawa [Wot09] differs from this approach by computing all possible diagnoses and then using additional observations to discriminate the diagnoses until—preferably—a single solution is left. The discrimination process uses the same idea as de Kleer and Williams [DKW87], who take advantage of entropy to select the next observation point. Choosing an observation that is predicted by only one half of the diagnoses, allows us to split the search space. de Kleer and Williams apply this in the context of the traditional consistency-based approach and consider all diagnoses candidates, because any superset of a consistency-based diagnosis candidate is a diagnosis itself. Notice that this property only applies to approaches using models of the correct system behavior. Hence, this does not hold for the abductive case, where a formalization of specific failures is required. Consequently in our case only minimal abductive diagnoses are further investigated.

A discriminating observation is a measurement not yet considered, which decreases the number of possible faults.

Definition 3.5 (Discriminating Observation [Wot09]). Given a *PHCAP* (*A*,*Hyp*,*Th*,*Obs*) and two diagnoses Δ_1 and Δ_2. A new observation $o \in A \setminus Obs$ discriminates two diagnoses if and only if Δ_1 is a diagnosis for (*A*,*Hyp*,*Th*,*Obs* $\cup \{o\}$) but Δ_2 is not.

According to information theory, the observation with the highest entropy $H(o)$ (defined in Equation (3.9)) provides the best probing point [DKW87].

$$H(o) = -p(o) \cdot \log_2 p(o) - (1 - p(o)) \cdot \log_2(1 - p(o)) \tag{3.9}$$

$p(o)$ denotes the probability of observation o and is defined as:

$$p(o) = \frac{|\{\Delta | \Delta \in \Delta\text{-Set}, \Delta \cup Th \models \{o\}\}|}{|\Delta\text{-Set}|} . \tag{3.10}$$

Once the next best probing point has been selected and the additional measurements have been taken, the probing results are passed on to the diagnosis engine as observations and the fault identification process is restarted. Ideally, the diagnosis engine reuses the diagnoses computed before and only considers this new information. Within the *PHCAP* a diagnosis Δ is discriminated in case we observe a complementary observation, i.e., $\neg o \subseteq Obs : \Delta \cup Th \models \{o\}$.

3.4.2 Fault Ranking

According to Bayes rule the conditional probability of an explanation can be computed as:

$$p(\Delta \mid o) = \frac{p(o \mid \Delta) p(\Delta)}{p(o)} \tag{3.11}$$

Since we aim at determining the probability of a diagnosis given the *PHCAP*, we only consider effects actually observed. Let us further assume that there is no uncertainty in the measurement, hence the data has not be subjected to errors or noise. For any $o \in Obs$, we can infer that $p(o) = 1$. As the explanation logically implies the observation we can assign $p(o \mid \Delta) = 1$. Hence, $p(\Delta \mid o) = p(\Delta)$. Assuming independence amongst faults, the probability of each diagnosis Δ can be computed based on the a-priori probability[5] $p(h)$ of each hypothesis h represented within the diagnosis, as shown in Equation (3.12).

$$p(\Delta) = \prod_{h \in \Delta} p(h) \prod_{h \notin \Delta} (1 - p(h)) \tag{3.12}$$

Given a *PHCAP* we compute $p(\Delta)$ for all diagnoses in Δ-Set and subsequently assign ranks accordingly. Besides using a probabilistic approach, to define an ordering for diagnoses, the failure assessments provide additional information that may be useful for determining a priority. For instance, FMEA contains a severity rating that quantities the notion of seriousness of the failure's consequences and would allow to rank solutions according to gravity of their effects.

3.5 Conclusion

In the course of the presented research, we proposed a process to facilitate the adoption of abductive model-based diagnosis in industrial practice. The offline part of the approach automatically generates diagnosis models from failure assessment documents, thus, diminishing

[5]The a-priori probabilities may be available from manufacturers, however, in practice more often no such information is accessible. Hence, this knowledge has to be obtained first, e.g., by fault history analysis.

to completely avoiding the initial modeling effort hindering a practical application of model-based diagnosis. The online portion requires a fault detection mechanism and identifies a failure through a cycle of diagnosis, ranking, and probing. We discuss two different analysis methods, namely FMEA and FTA, in regard to their capabilities to form the basis of an abductive diagnosis process. These assessments contain information suitable for abductive system descriptions and allow us to automatically build models offline. Exploiting failure assessments is a feasible approach, as on the one hand this sort of analyses is becoming increasingly important and on the other hand it diminishes the modeling effort required.

Depending on the utilized assessment, the resulting models differ in their structure and expressiveness. The Horn theory constructed from the information stored within an FMEA is the most straightforward and simple, as each clause represents a causal relation between a single failure and a single effect. Due to the consequences of a cause being related by conjunctions, the produced models are rather strict in the sense that in case the opposite of one manifestation out of the effect set is perceived, the corresponding cause is no longer a viable part of a diagnosis. Thus, the associated hypotheses have to be removed from all solutions. In practice, this signifies that observation data have to be preprocessed to remove noise or account for inaccurate measurements to ensure that feasible fault modes are not discriminated prematurely. As the FMEA comprises certain information on occurrence likelihood and severity or criticality, additional improvements to the diagnosis results are possible. Ideally, a-priori failure probabilities are known from manufacturers or historic data. However, even ratings, as are very commonly used in FMEA, can indicate a prioritization. Particularly severity or criticality can signify failures most paramount in regard to safety or economic considerations. From a practical and research point of view interesting characteristic of the FMEA models is that they are bijunctive definite Horn clauses and hence abduction is tractable on these system descriptions.

In comparison to the FMEA-based models, fault trees can express a wider range of situations by allowing to represent the combination of events. To create a *KB* for an entire system, the modeling requires a fault tree for each possible observation and subsequently the knowledge bases have to be joined in a comprehensive system description. In case the FTA is quantitative, the probabilities of the basic events corresponding to hypotheses can be incorporated to determine cause likelihoods. Interesting enough to create abductive diagnoses a mapping to a *KB* is not necessary, as we could exploit the notion of minimal cut sets. Assume a *PHCAP* with *Obs* $= \{o_1, \ldots, o_n\}$ and for each $o_i \in$ *Obs* there is a fault tree $ft_i \in \mathbb{T}$ with ε corresponding to o_i. To obtain a minimal cut set equivalent to the abductive explanations for the given problem, the fault trees need to be joined. A combined fault tree with a new top event *ex* is introduced with a gate $\alpha \in \mathcal{A}(I, \omega)$ such that $\omega = \{ex\}$ and $I = \bigcup_{ft_i \in \mathbb{T}} \varepsilon$. The minimal cut sets of *ex* then constitute the prime implicants or minimal abductive diagnoses of the *PHCAP*.

Essentially, the advantages and disadvantages of the underlying assessments are not only inherent in the type of relation they are capable of expressing, but also in the incorporated additional information they hold. In particular, any ranking information can be viable to determine probable or critical diagnoses. It is apparent that the quality of the model automatically generated is largely depended on the underlying failure assessment. Failure modes or effects not considered in the analysis, are absent in the abductive model and thus diagnoses involving those cannot be uncovered. Model completeness is a primary requirement, thus, an essential aspect is a systematic and comprehensive review of the system to achieve a high coverage of faults and their consequences [Mil+00]. Certain assumptions are fundamental to our approach, e.g., to ensure the feasibility of an automatic creation of the model, manifestations and failures have to be coherently reported throughout

the assessment and consequences have to be detectable in order to be useful in a diagnostic context.

4 Solving Bipartite Diagnosis Problems

> *That we find out the cause of this effect,*
> *Or rather say, the cause of this defect,*
> *For this effect defective comes by cause.*
>
> — **William Shakespeare**
> "*Hamlet*". around 1602.

This chapter is based on the following publications:

- [KW15a] Roxane Koitz and Franz Wotawa. „Finding Explanations: An Empirical Evaluation of Abductive Diagnosis Algorithms“. In: *Proceedings of the 2015 International Workshop on Defeasible and Ampliative Reasoning*. CEUR-WS. org. 2015, pp. 36–42
- [KW15d] Roxane Koitz and Franz Wotawa. „SAT-Based Abductive Diagnosis“. In: *26th International Workshop on Principles of Diagnosis*. 2015, pp. 1–9
- [KW16a] Roxane Koitz and Franz Wotawa. „Improving Abductive Diagnosis Through Structural Features: A Meta-Approach“. In: *Proceedings of the 2016 International Workshop on Defeasible and Ampliative Reasoning*. 2016, pp. 1–9
- [KW16c] Roxane Koitz and Franz Wotawa. „On Structural Properties to Improve FMEA-Based Abductive Diagnosis“. In: *Proceedings of the Workshop on Knowledge-based Techniques for Problem Solving and Reasoning*. 2016, pp. 1–7

The examination and empirical evaluation of conflict-driven methods for computing explanations have been published in [KW15d]. In [KW15a], we investigate the equivalence between set covers and hitting sets to formulate the solutions to a bipartite abduction problem based on hitting set computation. The benchmarks for the first experiment have been published in [KW16c] and we conducted the second experiment allowing to solve diagnosis on the fly with experiment data from [KW16a].

4.1 Motivation

In the previous chapter, we have introduced a general process for applying fault identification to technical systems exploiting a model-based diagnosis approach. Even though based on a well-defined theory, a widespread acceptance of model-based diagnosis among industries has not been accounted for yet. The initial model development and the computational complexity of diagnostic reasoning are some of the main reasons for this observation. In order to diminish the modeling effort, we have formulated a conversion of failure assessments available in practice into a propositional logic representation suitable for abductive diagnosis. Our first information source were FMEAs. FMEA is an established reliability evaluation method utilized in various industrial fields that considers possible component faults as well as their implications on the system's behavior. Thus, this type of analysis poses an appropriate base for automatically extracting abductive diagnosis models. Regarding the second issue associated with model-based diagnosis, i.e., its computational effort, we have seen that the models built on top of FMEAs observe a certain structure that leads to a diminished complexity in comparison to general propositional Horn abduction. Each clause

constructed from an FMEA is a bijunctive definite Horn clause, meaning a single hypothesis is linked to a single manifestation via an implication[1].

We have provided some initial empirical evaluation using an abductive procedure exploiting an ATMS. The ATMS propagates the information of assumptions through an And-or-graph that represents the underlying Horn theory. The models based on FMEAs do not feature various levels or depths within in this graph, but are shallow, i.e., there is exactly two levels. In addition, there are only two types of entities, hypotheses and effects, and each directed edge leads from one hypothesis to one effect. Hence, the theory can be represented as a bipartite graph. Recalling the preliminaries, we have introduced the parsimonious set covering approach. In its simplest form, the causal associative network is a bipartite graph. Hence, the bijunctive definite Horn clause theory is equivalent to the bipartite abduction problems in the simple set-covering theory. This simple model structure suggests that a simpler procedure might also suffice in computing the explanations. Based on the equivalence of the simple set-cover problem and abduction based on bijunctive definite Horn clause, we can utilize the relation of set-covering and hitting set computation to exploit hitting set procedures to extract diagnoses.

In this chapter, we provide two general directions for solving these bipartite abduction problems. On the one hand, we evaluate proof-tree completion [McI98] as a method to compute diagnoses for these simple abduction problems. Thus, in Section 4.2 we recall the basics of conflict-driven abduction and subsequently present four procedures to extract minimal explanations within this framework. On the other hand, we rely on the framework of the parsimonious set-covering theory as proposed by Peng and Reggia [PR90]. Viewing the simple abduction problems based on FMEAs as set cover problems allows us to compute explanations using various hitting set algorithms. In particular, we present five hitting set approaches and compare them to abduction using the ATMS.

4.2 Conflict Driven Techniques

The computation of explanations has not only been studied in the context of diagnosis, but also has received attention in the field of constraint satisfaction problems and infeasibility analysis. Junker [Jun04] describes an algorithm generating preferred explanations for over-constrained systems. By employing a divide and conquer strategy, conflicting constraints can be efficiently computed. These contradictions essentially constitute the causes for the unsatisfiability of the system. Hence, we rely on this connection between conflicts and abductive explanations that we have already discussed in the preliminaries, i.e., proof-tree completion [McI98]. Recall that in proof-tree completion, we rely on a reformulation of the abduction problem in such a way that a conflict arises given the observations; $\Delta \cup Th \cup \neg\{Obs\} \models \bot$ where $\neg\{Obs\}$ is a disjunction of the negated literals of the observations, i.e., $\bigvee_{o \in Obs} \neg o$.

Example 4.1

Consider the following *PHCAP* encompassing a definite Horn theory:

$$A = \{H_1, H_2, H_3, o_1, o_2, o_3\}, Hyp = \{H_1, H_2, H_3\},$$

$$Th = \left\{ \; H_1 \rightarrow o_1, H_2 \rightarrow o_1, H_2 \rightarrow o_2, H_3 \rightarrow o_2, H_3 \rightarrow o_3 \; \right\}, Obs = \{o_1\}$$

[1] Note here that we ignore additional rules that may be applied to state impossibilities, i.e., an observation and its complement cannot occur at the same time.

Assuming all hypotheses are true, while stating $\neg o_1$, conflicts arise. Just consider the first clause in *Th* defining that if H_1 is present o_1 is observable. Now stating that H_1 is true and o_1 is false is contradicting *Th* thus $\{H_1\} \cup Th \cup \neg o_1$ is inconsistent. Extracting only the hypotheses we retrieve the first minimal conflict $\mathcal{CO}_1 = \{H_1\}$. Since the conflict is consistent with the theory *Th*, $\mathcal{CO}_1$ is a minimal diagnosis. The second conflict, i.e., solution, $\mathcal{CO}_2 = \{H_2\}$ is derived analogously.

4.2.1 Minimal Unsatisfiable Subset and Minimal Correction Subset

The advances in the development of SAT solvers and their application to a vast number of different AI problems and industrial domains have motivated us to consider a SAT-based approach for abductive diagnosis. Given a formula ϕ, the decision problem SAT consists of deciding whether there is a satisfying assignment for the formula. The performance of SAT solvers has improved immensely over the last years and several applications of SAT solvers in practice have proven successful. Furthermore, we are able to encode a greater variety of models in SAT.

Within the context of infeasibility analysis conflicts are known as Minimal Unsatisfiable Subsets (MUSes) or unsatisfiable cores. In case ϕ is unsatisfiable, an MUS is a set of clauses that cannot be satisfied simultaneously, while every proper subset of an MUS is satisfiable [LS08]. Minimal Correction Subsets (MCSes) are the clauses that corrects the unsatisfiable formula, i.e., by removing any MCS the formula becomes satisfiable. MUSes can be computed either directly or via enumerating their hitting set dual MCS and subsequently deriving all irreducible hitting sets [Rei87].

Definition 4.1 (Minimal Unsatisfiable Subset (MUS) [LS08]). Given an unsatisfiable propositional formula ϕ, a subset $U \subseteq \phi$ is an MUS if U is unsatisfiable and $\forall C_i \in U, U \setminus \{C_i\}$ is satisfiable.

Definition 4.2 (Minimal Correction Subset (MCS) [LS08]). Given an unsatisfiable propositional formula ϕ, a subset $M \subseteq \phi$ is an MCS if $\phi \setminus M$ is satisfiable and $\forall C_i \in M, \phi \setminus (M \setminus \{C_i\})$ is unsatisfiable.

An MCS contains clauses that correct the unsatisfiable formula when removed and can be defined through the set of Maximal Satisfiable Subsets (MSSes), since each MCS is the complement of a MSS.

Definition 4.3 (Maximal Satisfiable Subset (MSS) [LS08]). Given an unsatisfiable propositional formula ϕ, a subset $S \subseteq \phi$ is an MSS if S is satisfiable and $\forall C_i \in (C \setminus S), S \cup \{C_i\}$ is unsatisfiable.

Example 4.2

Consider the unsatisfiable formula

$$\phi = \overbrace{(\neg x \vee \neg y)}^{C_1} \wedge \overbrace{(z)}^{C_2} \wedge \overbrace{(\neg z \vee y)}^{C_3} \wedge \overbrace{(\neg y)}^{C_4} .$$

The combination of clauses C_2, C_3 and C_4 results in ϕ being unsatisfiable; hence, $\text{MUSes}(\phi) = \{\{C_2, C_3, C_4\}\}$. By hitting set computation we derive the follow-

ing MCSes(ϕ) $= \{\{C_2\},\{C_3\},\{C_4\}\}$ and subsequently can determine MSSes(ϕ) $= \{\{C_1,C_3,C_4\},\{C_1,C_2,C_4\},\{C_1,C_2,C_3\}\}$.

Many algorithms for computing unsatisfiable cores, however, do not generate them directly, but rely on their hitting set dual MCSes. Liffiton and Sakallah [LS08] propose the CAMUS algorithm utilizing this hitting set duality to produce MUSes by first computing all MCSes, since in practice finding MUSes directly is more challenging than extracting MCSes. Recently, Liffiton et al. [Lif+16] have presented a direct MUSes extraction approach that exploits the power-set lattice. Utilizing subsets of unsatisfiable formulas has been suggested in regard to consistency-based diagnosis [Rei87; MS+15]. In this context, a diagnosis is defined as the set of components which assumed faulty retains the consistency of the system. Thus, a consistency-based diagnosis corresponds to an MCS. For instance, Felfernig and Schubert [FS10] present a direct consistency-based diagnosis method computing MCSes for over-constrained systems. In our abductive case, yet, we aim at extracting contradictions, i.e., MUSes, from an unsatisfiable system description. Since MUSes contain several unsatisfiable subsets irrelevant for the diagnostic task, we define the set MUSes$_{Hyp}$, which only contains subset minimal MUS comprising clauses referring to hypotheses, i.e., our abducible propositions.

Definition 4.4 (MUSes$_{Hyp}$). Let MUSes be the set of MUSes of $Hyp \cup Th \cup \{\neg Obs\}$, then $\forall M \in \text{MUSes}_{Hyp} : \exists U \in \text{MUSes} : M = U \cap Hyp$ and $\neg\exists M' \in \text{MUSes}_{Hyp} : M' \subset M$.

Corollary 1. Given a *PHCAP*(*A*,*Hyp*,*Th*,*Obs*), let MUSes$_{Hyp}$ be the set of interesting MUSes. A set $\Delta \subseteq Hyp$ is a minimal abductive diagnosis if $\exists M \in \text{MUSes}_{Hyp} : \Delta = M$ and $\Delta \cup Th \not\models \bot$.

Proof. We can restate the problem of computing inconsistencies to finding the set of prime implicates of $Th \wedge Hyp \wedge \{\neg Obs\}$. By definition, the prime implicates are equivalent to the MUSes of said formula. □

Thus, deriving a minimal abductive explanation corresponds to computing a minimal subset of the hypotheses, which cannot be simultaneously satisfied with the theory and the negation of observations.

Indirect Approach

We devised the algorithm SATAB that computes the set of abductive diagnoses for a given *PHCAP* based on MUS enumeration. First, in order to take advantage of the MUSes, which correspond to the solutions of the *PHCAP*, we create an unsatisfiable CNF encoding of the problem. Since the *Th* consists of Horn clauses a conversion into CNF is straightforward. Note that we are, however, not limited to Horn clause models, as we can create a CNF representation based on Tseitin transformation [Tse70]. We refer to the set of clauses associated with the theory as $\mathcal{T}$. For each $h \in Hyp$ we create a single clause assuming h to be true. Additionally, we generate a disjunction containing the negated observations.

Algorithm 4.1: SATAB

Require: *A*: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: the Horn clause theory, *Obs*: the set of observations
Ensure: Δ-Set: set of minimal diagnoses
1: $MCSes \leftarrow \emptyset$
2: $MCSes_{Hyp} \leftarrow \emptyset$
3: $\mathcal{T} \leftarrow$ `CNF(Th)` ▷ CNF representation of *Th*

```
4: φ ← T ∪ Hyp ∪ ⋁_{o∈Obs} ¬o
5: MCSes ← MCSes(φ)                          ▷ MCS enumeration algorithm
6: for all M ∈ MCSes do
7:     if M ⊆ Hyp then                       ▷ MCS consisting of elements of Hyp
8:         for all m ∈ M do
9:             if m ∪ Th is inconsistent then
10:                goto L1
11:            end if
12:        end for
13:        MCSes_Hyp ← M ∪ MCSes_Hyp
14:        L1
15:    end if
16: end for
17: Δ-Set ← MHS(MCSes_Hyp)                   ▷ Minimal hitting set algorithm
18: removeInconsistent(Δ-Set)                ▷ Removes any inconsistent solutions
19: return Δ-Set
```

The diagnostic task consists in computing the sets of hypotheses which are responsible for the unsatisfiability of ϕ, i.e. $\text{MUSes}_{Hyp}(\phi)$. We employ an MCSes enumeration algorithm on the unsatisfiable formula and then derive the diagnoses via hitting set computation [LS08]. As we are only interested in the conflicts stemming from the assumptions that all hypotheses are true, we select each MCS only containing clauses referring to explanations. For this reason, we create the set MCSes_{Hyp} such that $\forall m \in \text{MCSes}_{Hyp} : m \subseteq Hyp$. In addition, this has a practical rational: it diminishes the number of sets to be considered by the hitting set algorithm. The corresponding MUSes derived via hitting set computation of MCSes_{Hyp} already constitute the abductive diagnoses.

Example 4.1 (cont.)

The CNF representation ϕ of the abduction problem looks as follows:

$$\phi = \overbrace{(\neg H_1 \vee o_1)}^{C_1} \wedge \overbrace{(\neg H_2 \vee o_1)}^{C_2} \wedge \overbrace{(\neg H_2 \vee o_2)}^{C_3} \wedge \overbrace{(\neg H_3 \vee o_2)}^{C_4} \wedge \overbrace{(\neg H_3 \vee o_3)}^{C_5} \wedge \overbrace{H_1}^{C_6} \wedge \overbrace{H_2}^{C_7} \wedge \overbrace{H_3}^{C_8} \wedge \overbrace{\neg o_1}^{C_9}$$

Clauses C_1 to C_5 correspond to $\mathcal{T}$, while C_6 to C_8 ensure that the hypotheses are assumed true. Lastly, the observation o_1 is added as its negation in clause C_9. Computing the MCSes of ϕ we obtain $\text{MCSes} = \{\{C_1, C_2\}, \{C_6, C_7\}, \{C_9\}\}$ Extracting the MCSes, which only contain clauses from *Hyp* and are consistent with regard to the theory, results in $\text{MCSes}_{Hyp} = \{\{C_6, C_7\}\}$. By computing the hitting set of MCSes_{Hyp}, we obtain the set of MUSes solely referring to explanations, which is in fact the set of diagnoses $\Delta\text{-Set} = \{\{H_1\}, \{H_2\}\}$.

Direct Approach

Algorithm MUSAB employs a MUS enumeration procedure and thereon computes the minimal abductive diagnoses, denoted $MUSes_{Hyp}$. Again we create an unsatisfiable CNF encoding of the problem denoted ϕ. The MUSes returned by the MUS enumeration method have to be stripped from all propositions not representing a hypothesis. The resulting set

denoted M may not be a minimal set of sets, i.e., there may be supersets. These supersets are not copied to $MUSes_{Hyp}$.

Algorithm 4.2: MUSAB

Require: A: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: the Horn clause theory, *Obs*: the set of observations
Ensure: Δ-Set: set of minimal diagnoses

```
MUSes ← ∅
MUSes_Hyp ← ∅
M ← ∅
𝒯 ← CNF(Th)                                   ▷ CNF representation of Th
ϕ ← 𝒯 ∪ Hyp ∪ ⋁_{o∈Obs} ¬o
MUSes ← MUSes(ϕ)                              ▷ MUS enumeration algorithm
for all ϒ ∈ MUSes do                          ▷ Remove propositions not in Hyp
    M ← ϒ ∩ Hyp
end for
for all ϒ ∈ M do
    if ∄ϒ′ ∈ M : ϒ′ ⊆ ϒ and ϒ ∪ Th is consistent then
        MUSes_Hyp ← ϒ                         ▷ Disregard supersets and inconsistent solutions
    end if
end for
Δ-Set ← MUSes_Hyp
return Δ-Set
```

Example 4.1 (cont.)

In MUSAB the same CNF representation of the diagnosis problem is used. We obtain the following MUSes from ϕ: $MUSes = \{\{C_1, C_6, C_9\}, \{C_1, C_7, C_9\}, \{C_2, C_6, C_9\}, \{C_2, C_7, C_9\}\}$. Since we are only interested in the abducibles, we remove all clauses not associated with hypotheses. Let $M = \{\{C_6\}, \{C_7\}\}$ be the resulting set. In case there are any supersets within M those have to be removed, resulting in $MUSes_{Hyp} = \{\{C_6\}, \{C_7\}\}$ and hence $\Delta\text{-Set} = \{\{H_1\}, \{H_2\}\}$

4.2.2 Conflict-Driven Search via HS-DAG

By detecting a discrepancy between the predicted and actual behavior, i.e., a conflict, Reiter [Rei87] derives consistency-based diagnoses via first identifying conflicts and subsequently computing the diagnosis via minimal hitting set computation. Algorithm HSDAGAB is based on Hitting Set Directed Acyclic Graph (HS-DAG) and a theorem prover to derive conflicts, which in our scenario can be used to subsequently extract the minimal abductive explanations. Given a *PHCAP*, we generate an implication with a conjunction of observations on the left hand side and the contradiction on the right hand side, i.e. $o_1 \wedge o_2 \wedge \ldots \wedge o_n \rightarrow \bot$[2]. The theory *Th*, the implication, and the theorem prover, represented by *TP*, are supplied to HS-DAG. The *TP* derives contradictions on demand, which should only contain assumption, i.e., hypotheses. In particular, whenever a new node n is appended to the DAG, it is checked for consistency using *TP*, i.e., the theorem prover is invoked assuming that all assumptions except the ones in $h(n)$ are true. In the case of an inconsistency arising, we are storing it in *CONF* and ensure that *CONF* is consistent with the background theory *Th*. Note that

[2]The exact encoding is dependent on the applied theorem prover.

depending on the utilized theorem prover these conflicts are not guaranteeing to be minimal; thus, we remove all supersets afterwards.

Algorithm 4.3: HSDAGAB

Require: A: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: the Horn clause theory, *Obs*: the set of observations
Ensure: Δ-Set: set of minimal diagnoses

```
1:  CONF ← ∅
2:  TP ← Th ∪ (⋀_{o∈Obs} o → ⊥)                ▷ Theorem Prover
3:  CONF ← HS-DAG(TP)                          ▷ Conflicts derived during HS-DAG construction
4:  removeInconsistent(CONF)                   ▷ Removes any inconsistent conflicts
5:  for all c ∈ CONF do
6:      if ∄c' ∈ CONF : c' ⊆ c then            ▷ Remove conflict supersets
7:          Δ-Set ← c
8:      end if
9:  end for
10: return Δ-Set
```

Example 4.1 (cont.)

Starting from the root of the HS-DAG in Figure 4.1, the first consistency check is performed, assuming all hypotheses are true as $h(root) = \emptyset$. We can obtain the first contradiction from *TP*, i.e., $\mathcal{CO}_1 = H_1$ which is added to the set *CONF*. Since the conflict has a cardinality of 1, only one new node n_1 is created, such that its edge label is the union of the root's edge label and the conflict element, i.e., $H(n_1) = \emptyset \cup \{H_1\}$. In the next step the theorem prover is called assuming H_2 and H_3 true, and H_1 false leading to $\mathcal{CO}_2 = \{H_2\}$ (see Figure 4.2). The consistency check for n_2 determines that there are no more contradictions, the node is marked ✓ and since there are no more nodes to process the computation is finished (see Figure 4.3). Since the derived conflicts are consistent with the background theory and subset minimal, they constitute the diagnoses, i.e., *CONF* = Δ-Set = $\{\{H_1\}, \{H_2\}\}$.

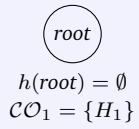


Figure 4.1: Initial DAG.

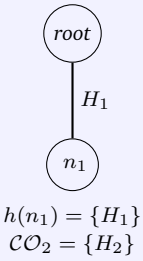


Figure 4.2: First level. *CONF* = $\{\{H_1\}\}$

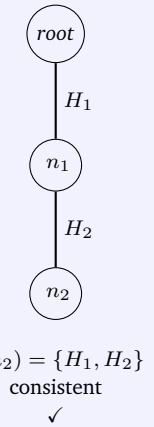


Figure 4.3: Second level.*CONF* = $\{\{H_1\}, \{H_2\}\}$

4.2.3 Empirical Evaluation

In this section, we describe our empirical evaluation set-up and report on the obtained results. All the numbers presented in this section were obtained from a Lenovo ThinkPad T540p Intel Core i7-4700MQ processor (2.60 GHz) with 8 GB RAM running Ubunutu 14.04 (64-bit).

Using the Indirect Approach

To determine whether computing abductive diagnoses via SAT yields any computational advantages in the case of our models, we conducted an empirical evaluation, comparing the ATMS referred to as Algorithm ABDUCTIVEEXPLANATIONS to SATAB on several new problem instances based on the FMEAs (in their original and OSFDP variant) we have utilized in the previous evaluations. In case of the former we again employed a Java implementation of an unfocused ATMS. The algorithm SATAB exploits on the one hand an MCS enumeration procedure and on the other hand an implementation of a hitting set algorithm. We utilized the MCS_{LS}[3] tool by Marques et al. [MS+13] to compute the MCSes. MCS_{LS} is written in C++, employs Minsat 2.2[4] as the SAT solver, and provides the possibility to apply several MCS enumeration algorithms. We decided for the CLD approach of MCS_{LS}, which takes advantage of disjoint unsatisfiable cores and showed the best overall performance in a preliminary experimental set-up. Regarding the hitting set computation, we engaged a Java implementation of the Binary Hitting Set Tree (BHS-Tree) algorithm [LJ03] which performed well in a comparison of minimal hitting set algorithms [Pil+11].

Due to *Th* of an abductive knowledge base comprising Horn clauses, a conversion into a CNF representation, suitable for the MCS_{LS} tool, is straightforward. We do not address the model compilation times, since the system description would be compiled offline and the mapping execution consumed less than one second for the examples we utilized so far. Table 4.1 shows that none, except of the model resulting from the transformer's FMEA, of the original models satisfy the OSFDP. Therefore, we compiled a second set of models fulfilling the property by exchanging each set of indistinguishable hypotheses with a new single hypothesis representing said set. For example, Algorithm DISTINGUISHHYPOTHESES ensures that the resulting knowledge base satisfies the OSFDP. In Table 4.1 the original models are identified accordingly, and the adapted models are provided with the label OSFDP. Note that the number of hypotheses and rules diminishes for the adapted models.

In the experiments, we computed the abductive explanations for $|Obs|$ from one to the maximum number of effects possible. The observations were generated randomly; however, the same set was used for SATAB and ABDUCTIVEEXPLANATIONS on the original as well as adapted model. The results reported in Table 3.3 have been obtained from ten trials and both algorithms faced a 200 seconds runtime limit. Whereas some of the small runtimes are arguable due to the measurement in the milliseconds range, Table 3.3 reveals that SATAB (Mean = 703.73 ms, SD = 8432.07 ms, Median = 0.59 ms, Skewness = 18.61) does not outperform ABDUCTIVEEXPLANATIONS (Mean = 3.08 ms, SD = 16.38 ms, Median = 1 ms, Skewness = 12.68) in general. From the statistical data we can infer that the underlying distribution of both algorithms is highly right skewed, thus the bulk of values is located towards the lower runtimes. We can even observe that for certain instances, the SAT-based approach performs rather poorly. Amongst these are the model of an inverter and a rectifier of an industrial wind turbine. SATAB exceeded the given timeout four times for the former. Notice that in all these cases the MCSes generation already reached the time threshold. According to [MS+13] CLD requires $|\phi| - p + 1$ SAT solver calls, where p refers

[3] http://logos.ucd.ie/wiki/doku.php?id=mcsls
[4] http://minisat.se/

Component		#Diagnoses MAX	AVG	SF	DF	TF	Runtime [in ms] Algorithm	MIN	MAX	AVG
Electrical circuit	Original	792	197.15	11	11	66	**abductive Explanations**	< 1	425	27.87
							satAB	< 1	181.33	76.05
	OSFDP	1	1	1	1	1	**abductive Explanations**	< 1	8	0.33
							satAB	< 1	1.91	0.16
FCS	Original	18	2.93	3	6	18	**abductive Explanations**	< 1	1	0.42
							satAB	< 1	6.41	1.28
	OSFDP	18	2.75	3	6	18	**abductive Explanations**	< 1	61	2.04
							satAB	< 1	4.73	0.56
ACD	Original	15	2.89	5	15	15	**abductive Explanations**	< 1	84	1.38
							satAB	< 1	2.89	0.35
	OSFDP	10	2.04	5	10	10	**abductive Explanations**	< 1	1	0.29
							satAB	< 1	2.435	0.28
Main bearing	Original	3	2.54	3	0	0	**abductive Explanations**	< 1	1	0.16
							satAB	< 1	1	0.09
	OSFDP	2	1.54	2	0	0	**abductive Explanations**	< 1	1	0.12
							satAB	< 1	0.61	0.03
HIFI - FPU	Original	63	8.64	3	7	21	**abductive Explanations**	< 1	86	2.54
							satAB	< 1	8.33	3.00
	OSFDP	6	1.55	2	2	3	**abductive Explanations**	< 1	1	0.15
							satAB	< 1	1	0.09
MiTS 1	Original	24	8.40	3	2	6	**abductive Explanations**	< 1	94	3.40
							satAB	< 1	3.02	0.39
	OSFDP	1	1	1	1	1	**abductive Explanations**	< 1	100	1.54
							satAB	< 1	2.15	0.16
MiTS 2	Original	288	39.98	4	8	18	**abductive Explanations**	< 1	109	4.49
							satAB	< 1	15.16	3.43
	OSFDP	5	2.02	1	5	2	**abductive Explanations**	< 1	1	0.33
							satAB	< 1	1.68	0.20
PCB	Original	2	1.49	2	2	2	**abductive Explanations**	< 1	1	0.21
							satAB	< 1	1.49	0.1
	OSFDP	1	1	1	1	1	**abductive Explanations**	< 1	1	0.11
							satAB	< 1	1	0.1
Inverter	Original	450	23.73	19	5	50	**abductive Explanations**	< 1	107	6.15
							satAB	< 1	166,593	5,007.37
	OSFDP	66	5.89	14	3	6	**abductive Explanations**	< 1	94	1.67
							satAB	< 1	1,110.82	38.23
Rectifier	Original	88	10.83	8	24	32	**abductive Explanations**	< 1	6	1.07
							satAB	< 1	2,4236.90	1,070.88
	OSFDP	22	3.06	5	18	8	**abductive Explanations**	< 1	1	0.63
							satAB	< 1	44.74	4.88
Trans-former	Original	2	1.06	2	2	1	**abductive Explanations**	< 1	1	0.16
							satAB	< 1	1.69	0.06
	OSFDP	2	1.06	2	2	1	**abductive Explanations**	< 1	1	0.13
							satAB	< 1	1.91	0.08
Backup	Original	252	23.06	8	12	21	**abductive Explanations**	< 1	138	5.24
							satAB	< 1	41.98	12.89
	OSFDP	48	3.29	7	7	10	**abductive Explanations**	< 1	4	0.79
							satAB	< 1	10.06	3.09

Table 4.1: Experimental results. For each component we conducted the experiment using an implementation of ABDUCTIVEEXPLANATIONS and SATAB. The columns *SF*, *DF*, *TF* display the maximum number of single faults, double faults, and triple faults, respectively.

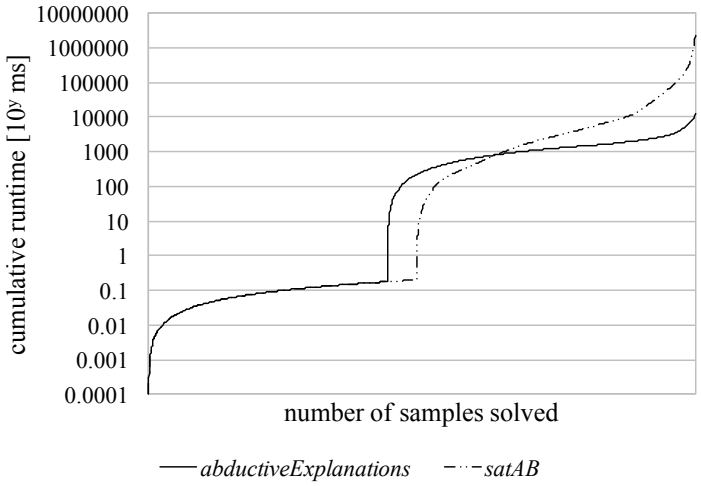


Figure 4.4: Cumulative runtimes of ABDUCTIVEEXPLANATIONS and SATAB for the FMEA instances.

to the size of the smallest MCS of ϕ. In our case $p = 1$, as the clause representing the set of negated observations always constitutes an MCS. Thus, $|\phi|$ SAT solver calls are necessary, where $|\phi|$ is determined by $|Th| + |Hyp| + 1$, with 1 referring to the clause containing the observations. Unsurprisingly, the larger FMEAs are more computationally demanding. It is worth mentioning that in the majority of cases the hitting set computation accounted for a negligible fraction of the total runtime.

Figure 4.4 illustrates the cumulative log runtimes for SATAB and ABDUCTIVEEXPLANATIONS on the FMEA models generated. ABDUCTIVEEXPLANATIONS performs on average better and Figure 4.4 reveals the high computational effort necessary for SATAB to compute the diagnoses. As expected we observe particularly high runtimes when the set of observations contains effects corresponding to different hypotheses. Generally, the data gathered in the experiment do not suggest a performance benefit of the SAT-based approach over an ATMS implementation.

Conflict-based Diagnosis

Besides the ATMS and the indirect approach SATAB, we utilize the HS-DAG to derive conflicts via HSDAGAB and the direct MUS approach MUSAB. For the former, we utilize the publicly available diagnosis engine jdiagengine that implements a conflict-driven search via HS-DAG [PW03] exploiting a Linear Time Horn Clause Theorem Prover (LTUR) [Min88]. jdiagengine as well as HSDAGAB are Java implementations. We implemented MUSAB in Java and employed the MUS enumeration tool MARCO[5] [Lif+16]. MARCO computes MUSes and MSSes based on an exploration of the power-set lattice[6]. Given an unsatisfiable clause set, all of its supersets are unsatisfiable as well; thus, an MUS defines a "low point" in an infeasible region. Similarly, an MSS characterizes a "high point" in a satisfiable region. In each iteration MARCO investigates an unexplored part of the lattice and traverses through the power-set until either an MUS or an MSS is found. MARCO is implemented in Python using the MUS extractor MUSer2[7] and SAT solver MiniSat[8].

For this experiment, we generate new *PHCAP*s based on the FMEAs. Table 4.2 provides an overview of the models' structure as well as some characteristics of the problem instances.

[5] `http://sun.iwu.edu/~mliffito/marco/`
[6] We provide a more detailed description of this algorithm in Subsection 5.2.4.
[7] `http://logos.ucd.ie/wiki/doku.php?id=muser`
[8] `http://minisat.se/`

Model	Structure			# Diagnoses				
	Hyp	*Obs*	*Th*	Max	Avg	SF	DF	TF
Electrical circuit	32	17	52	792	189.10	3	3	12
Ford connector system	17	17	56	28	3.68	5	3	15
ACD	13	16	52	12	2.46	3	4	4
Main bearing	3	5	20	3	2.34	3	0	0
HIFI-FPU	17	11	35	42	9.44	7	21	7
MiTS 1	17	21	47	12	5.04	3	3	4
MiTS 2	22	15	48	288	33.46	4	12	6
PCB	10	11	24	2	1.52	1	2	2
Inverter	29	38	165	200	21.79	2	14	16
Rectifier	20	17	93	64	8.10	16	32	64
Transformer	4	8	22	2	1.10	2	2	2
Backup components	25	30	114	252	19.86	7	18	27

Table 4.2: Features of the models and the evaluation examples. *SF*, *DF*, and *TF* refer to single, double, and triple faults, respectively.

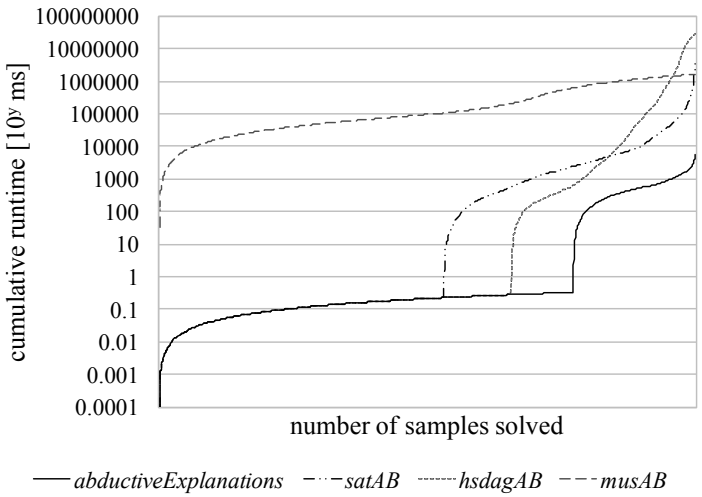


Figure 4.5: Cumulative runtimes of ABDUCTIVEEXPLANATIONS, HSDAGAB, MUSAB, and SATAB for the experiment.

Model	abductive Explanations		hsdagAB		satAB		musAB		
	MAX	AVG	MAX	AVG	MAX	AVG	MIN	MAX	AVG
Electrical circuit	129	19.44	**T**	5,131.66	145.05	51.49	8	6,881	700.06
FCS	5	0.23	18	1.22	5.12	0.78	7	1,974	419.95
ACD	12	0.28	3	0.31	7.00	0.34	7	122	42.28
Main bearing	1	0.02	1	0.04	1.00	0.07	11	269	93.86
HIFI - FPU	1	0.04	174	9.42	6.05	1.98	7	469	141.82
MiTS 1	1	0.09	1	0.10	2.42	0.23	7	37	19.1
MiTS 2	12	0.57	164,891	3,522.88	11.00	2.53	7	5,281	905.45
PCB	1	0.01	1	0.01	1.99	0.12	7	12	9.36
Inverter	55	2.62	**T**	15,406.82	**T**	3,799.10	8	**T**	14,573.4
Rectifier	4	0.32	25,830	233.51	11,450.60	455.11	8	**T**	17,173.03
Transformer	1	0.01	< 0.001	< 0.001	0.73	0.04	7	36	19.63
Backup	25	2.03	**T**	4,113.47	35.84	9.69	8	**T**	14,526.67

Table 4.3: Experimental runtime [in ms] results of the four algorithms on the experiment instances. Models, where an algorithm exceed the given run time threshold at least once, are marked with **T**.We only include the minimum runtime **MIN** for MUSAB since it is the only algorithm with minimal runtimes > 1.

Due to theory comprising Horn clauses, a conversion into a CNF representation, suitable for the MUS-based and MCS-based computation, is straightforward. In comparison to the previous evaluation, we do not consider the OSFDP property. In the experiments, we computed the abductive explanations for $|Obs|$ from one to the maximum number of effects possible with the observations selected randomly. The results reported in Table 4.3 have been obtained from ten trials and all algorithms faced a 200 seconds runtime limit. To compare the algorithms, we only measured the time to compute minimal diagnoses, i.e., we disregarded the mapping, model conversion, as well as the time it required to communicate with the solvers. In case of MUSAB and SATAB, we parsed the execution time measured by the tools themselves, which was available in the output. Note that for certain instances HSDAGAB, SATAB, and MUSAB exceeded the predefined runtime threshold, which we marked with **T** in the table. Thus, for the cumulative runtimes, shown in Figure 4.5, we utilized the maximum of 200 seconds in cases the limit was surpassed.

Whereas some of the small runtimes are arguable due to the measurement in the milliseconds range, Table 4.3 and Figure 4.5 reveal that ABDUCTIVEEXPLANATIONS (Mean = 2.41 ms, SD = 12.36 ms, Median = 0 ms) outperforms HSDAGAB (Mean = 12261.77 ms, SD = 3162.5 ms, Median = 1 ms), SATAB (Mean = 1741.39 ms, SD = 15633.06 ms, Median = 1 ms), and MUSAB (Mean = 45947.85 ms, SD = 82289.36 ms, Median = 118 ms). Unsurprisingly, the larger considered examples are more computationally demanding, especially with the model of the electrical circuit featuring a larger set of possible hypotheses and diagnoses. In cases where the maximum cardinality of the diagnoses is limited, HS-DAG computes solutions rather efficiently [Pil+11]. However, in our examples, we enumerated all solutions, thus neither the size nor the number of hitting sets was restricted, which can result in some cases in an extensive graph. The MCS-based approach performs rather poorly on the example of the converter. According to Marques et al. [MS+13] the number of SAT calls for the CLD approach depends on the size of the underlying formula, which in our case is determined by the size of the theory and the number of hypotheses, which explains the computation time for the inverter example. It is worth mentioning that in the majority of cases the hitting set computation accounted for a negligible fraction of the total runtime of SATAB. The performance of MARCO is very much dependent on the traversal of the graph towards a "low point" or "high point" in the power-set lattice, i.e., MUS or MSS, respectively. Note that we did not focus on an efficient encoding or any kind of pre-compilation to speed up the reasoning process. Further, in the case of MUS- and MCS-based algorithms, there is no focus on the abducibles, as for the ATMS and the HS-DAG. Thus, a large number of sets is generated, which are not of interest for the diagnostic task.

4.3 Abductive Diagnosis by Hitting Set Computation

Considering the structural properties of the models generated on basis of the FMEAs, we have stated that the models consist of bijunctive definite Horn clauses. Considering the implication-based representation of a Horn clause, a single hypothesis constitutes the premise of the rule, while a single variable representing an effect is the consequence, i.e., $H_i \rightarrow e_j$. In Peng and Reggia's [PR90] simple parsimonious set-covering theory the networks consist of a hypothesis and a manifestation layer, with the set of hypotheses and manifestations being disjoint. These types of abduction problems are referred to as *bipartite abduction problems*, since the causal network can be represented as a bipartite graph. Given this characteristic, this causal relation representation is equivalent to logic-based abduction with a theory restricted to bijunctive definite Horn clauses. Considering the definitions of the previous

chapter, $\mathcal{H}$ refers to *Hyp*, M is $A \setminus Hyp$ describing the remaining propositions not included within the hypotheses which are in fact the effects. The causal relations $\mathcal{T}$ corresponds to *Th* and $M^* \equiv Obs$.

Example 4.3

Consider the following *PHCAP* with $Obs = \{o_1, o_3\}$ and encompassing a bijunctive definite Horn theory:

$$A = \{H_1, H_2, H_3, o_1, o_2, o_3\}, Hyp = \{H_1, H_2, H_3\},$$

$$Th = \{\ H_1 \rightarrow o_1, H_2 \rightarrow o_1, H_2 \rightarrow o_2, H_3 \rightarrow o_2, H_3 \rightarrow o_3\ \}$$

We can formulate this within the simple set covering approach to a Set Cover Diagnosis Problem $\langle \mathcal{H}, \mathcal{M}, \mathcal{T}, \mathcal{M}^* \rangle$, where $\mathcal{H} = \{H_1, H_2, H_3\}$, $M = \{o_1, o_2, o_3\}$, $M^* = \{o_1, o_3\}$, and $\mathcal{T} = \{< H_1, o_1 >, < H_2, o_1 >, < H_2, o_2 >, < H_3, o_2 >, < H_3, o_3 >\}$.

As has been shown previously, set covering is equivalent to the hitting set problem [Aus+80]. Given a universal set $\mathbb{U}$ and a set of sets *CS*, set covering aims at identifying the minimal subsets C_i of *CS* such that their union covers $\mathbb{U}$, i.e., $\bigcup_{C_i \in CS} C_i = \mathbb{U}$. The hitting set problem is dual to set covering by exchanging $\mathbb{U}$ and *CS*, i.e., searching for subsets of $\mathbb{U}$ that cover the elements of *CS*. Formally, a hitting set is defined that given a set of sets *CS*, a hitting set h is a set intersecting all sets in *CS* with at least one element. h is said to be minimal if there exists no other hitting set h' for *CS* that is a subset of h [Pil+11].

Recall that in the set-covering theory the knowledge about the causal relations is defined by two sets, *effects*(h_i) and *causes*(m_j), and that a diagnosis is defined via a cover relation. That is, a set of hypotheses $\mathcal{H}'$ is a diagnosis if all observed manifestations are covered by $\mathcal{H}'$, i.e., $Obs \subseteq$ *effects*$(\mathcal{H}')$. A cover states that a certain disorder causally infers a manifestation, i.e., the manifestation $m \subseteq$ *effects*(H), where H is a hypothesis. Similarly, we can utilize the set *causes*(m_j) as previously defined as a similar cover indicator. For each manifestation the set *causes*(m_j) contains the information on all disorders causing m_j. Algorithm BIPARTITE as presented by Peng and Reggia [PR90], incrementally construct solutions covering a new manifestation in addition to the set of observations already considered using this *causes* relation. In essence, they incrementally derive hitting sets represented by generators. By computing the hitting set of *causes*(m_j) we derive a disjunction of all disorders included, i.e., each cause constitutes a possible solution. Hence, in case we obtain a set of observable manifestations $m_1, \ldots, m_n \in Obs$, the hitting sets of all *causes*$(m_j) \in Obs$ comprises the diagnoses. This is apparent, as to account for all current observations one disorder causing each manifestation has to be present within a single solution. Again we focus on parsimonious solutions, therefore we are solely interested in subset minimal hitting sets.

Definition 4.5 (Abductive Hitting Set Diagnosis). Given a bipartite abduction problem with *(A,Hyp,Th,Obs)*. A set $\Delta \subseteq Hyp$ is said to be a minimal diagnosis iff (1) Δ is a minimal hitting set of *CS*, where $\forall o_i \in Obs$: *causes*$(o_i) \in CS$, and (2) $\Delta \cup Th \not\models \bot$.

Example 4.3 (cont.)

Assuming we can observe o_1 and o_3, i.e. $Obs = \{o_1, o_3\}$, the *causes* sets for the current manifestations are *causes*$(o_1) = \{H_1, H_2\}$ and *causes*$(o_3) = \{H_3\}$. Thus, $CS = \{\{H_1, H_2\}, \{H_3\}\}$. The minimal hitting sets of *CS* are $\{H_1, H_3\}$ and $\{H_2, H_3\}$ hence correspond to $\Delta_1 = \{H_1, H_3\}$ and $\Delta_2 = \{H_2, H_3\}$.

Hence, we can utilize any algorithm capable of deriving all the minimal hitting sets for a given set *CS* to compute the diagnosis based in abduction problems constructed on top of an FMEA. Note, here however, that we have to take special care that the derived minimal hitting sets are consistent with the background theory. Similarly to the NOGOOD node in the ATMS, we could keep track of $causes(\bot)$.

Example 4.3 (cont.)

Assume the theory *Th* contains further a rule, stating that H_1 is contradicting, i.e.,

$$Th = \{\ H_1 \rightarrow o_1, H_2 \rightarrow o_1, H_2 \rightarrow o_2, H_3 \rightarrow o_2, H_3 \rightarrow o_3, H_1 \rightarrow \bot\ \}\ .$$

Given the same observations as before the derived minimal hitting sets are $\{H_1, H_3\}$ and $\{H_2, H_3\}$. The first hitting set, however, does not fulfill the second condition of Definition 4.5, since $\{H_1, H_3\} \cup Th$ is inconsistent. Hence, in this case the only admissible solution to the abduction problem is the second minimal hitting set $\{H_2, H_3\}$.

4.3.1 Hitting Set Algorithms

Given Definition 4.5 we can utilize hitting set algorithms to derive minimal abductive explanations for bipartite diagnosis problems. In this section, we present well-known techniques to derive minimal hitting sets. There is a growing body of literature on computing hitting sets not detailed in this chapter. For instance, STACCATO [AG09], which is based on the ideas of spectrum-based fault localization, provides an approximation of minimal hitting sets, while SAT-based hitting set techniques reformulate the problem as a CNF such that a satisfying assignment corresponds to the minimal hitting sets [Qua14]. For example, Pill, Quaritsch, and Wotawa [Pil+11] provide a detailed evaluation of minimal hitting set algorithms

Hitting Set Directed Acyclic Graph (HS-DAG)

By detecting a discrepancy between the predicted and actual behavior, i.e., a conflict, Reiter [Rei87] derives consistency-based diagnoses via minimal hitting set computation. A conflict arises when, under the assumption all components are behaving correctly, an observation is inconsistent with the expected performance. Thus, conflicts correspond to hypotheses contradicting observations in this context. Reiter's approach maintains a tree to compute all minimal hitting sets based on contradictions. These conflicts can be generated on demand by applying a theorem prover, which returns a refutation involving hypotheses if one exists. Starting from an initial conflict set as root node, the tree is iteratively extended in a breadth first manner. At each node n, labeled with conflict C, an outgoing edge $h(n)$ is generated for each $c \in C$. Each edge label is checked for consistency. In case it is consistent, the corresponding node determines a leaf and thus a minimal hitting set, otherwise a new conflict set is derived, such that it is disjoint to the current set of edge labels. Several pruning techniques ensure the minimality of the hitting sets and allow the use of non minimal conflicts. Greiner, Smith, and Wilkerson [Gre+89] corrected some inadequacies of Reiter's algorithm and devised an approach performing on a DAG instead of a tree referred to as HS-DAG. Algorithm HS-DAG lists their procedure.

Algorithm 4.4: **HS-DAG [Gre+89]**

1 Let *D* represent the growing DAG. Generate a node n_0 which will be the root of the DAG.

2 Process the nodes in *D* in a breadth-first order as long as there are open nodes. Define $h(n)$ to be the set of edge labels on the path in *D* from the root n_0 down to node n. Set $h(n_0) = \emptyset$.

- **a** (**Closing**) If there is a node n' which is labeled ✓ and $h(n') \subset h(n)$ then close node n, i.e., mark it ✗. The closed node n does not have any successors nor is a label computed for n.
- **b** If for all $C \in CS : C \cap h(n) \neq \emptyset$ then label n with ✓. Otherwise, label n by C where C is the first member of *CS* with $C \cap h(n) = \emptyset$.
- **c** (**Pruning**) If a previously unused C is to label a node and there is a node n' in *D* that has been labeled by $C' \in CS$ with $C \subset C'$, then
  - **i** Relabel n' with C and remove any arc labeled $c : c \in C' \setminus C$ under n'. The node connected by this arc and all of its descendants are removed, except for the nodes with another ancestor which is not being removed. Note that this step may eliminate the node which is currently being processed.
  - **ii** Interchange the sets C' and C in *CS* (this has the same effect as eliminating C' from *CS*).
- **d** If n is labeled by a set $C \in CS$, for each $c \in C$ generate a new downward arc labeled by c. If there is a node n' in *D* such that $h(n') = h(n) \cup c$, then let the c-arc under n point to this existing node n' (**Reuse**). Hence n' will have more than one parent. Otherwise let the arc c lead to a new node m with $h(m) = h(n) \cup \{c\}$. The new node m will be processed (labeled and expanded) after all nodes in the same generation (i.e., same depth) as n have been processed.

3 Return the resulting DAG *D*.

Example 4.4

Consider the set $CS = \{\{a, b\}, \{c, d\}, \{b, c\}, \{d\}\}$. Figure 4.6 depicts the constructed DAG before determining the next steps for the first node n_3 in the second level, i.e., the node with $h(n_3) = \{a, c\}$. The next conflict that could be used to expand the node is $\{d\}$, thus, the pruning rule takes effect. The nodes labeled with the conflict $\{c, d\}$ are labeled $\{d\}$ and the arcs leaving said node labeled c are no longer viable and hence removed. The final DAG is shown in Figure 4.7 with the resulting minimal hitting sets $\{a, c, d\}$ and $\{b, d\}$.

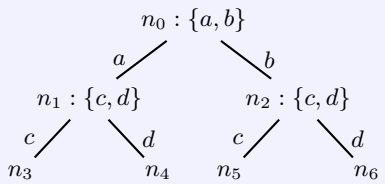


Figure 4.6: *D* before pruning.

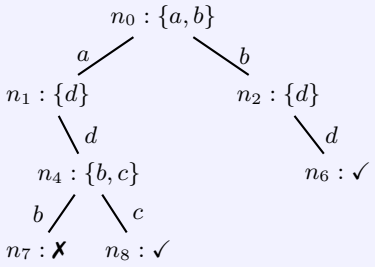


Figure 4.7: Final *D*.

Hitting Set Tree (HST)

The Hitting Set Tree (HST) variant of HS-DAG operates on a tree instead of a graph and avoids the construction of unnecessary nodes and costly subset checks [Wot01]. Based on an ordered list of elements within *CS* the algorithm limits the number of outgoing edges for each node to a specific range within the ordered list. Given this ordered list, the tree is constructed in a deterministic fashion allowing to decide whether a node has to be generated or the corresponding hitting set will be constructed later during the computation by checking the current path and the tree already constructed. This renders the reuse rule of the traditional HS-DAG unnecessary. Just as the HS-DAG, the conflict sets do not have to be known in advance, but may be derived on the fly by a theorem prover.

We define $\mathbb{U}$ as the universal set, i.e., $\forall C \in CS : C \subseteq \mathbb{U}$. HST relies on an algorithm enumerating all subsets of $\mathbb{U}$ in a tree structure by using a mapping function *ci* associating each element of $\mathbb{U}$ to a unique number between 1 and $|\mathbb{U}|$. The algorithm starts by setting the index i of the root node, representing the empty set, to $|\mathbb{U}| + 1$. For each number j between 1 and $i - 1$ a new vertex is constructed, such that the index of said node corresponds to j. A vertex is identified as a leaf in case it has an index of 1 otherwise the node's children are created.

Example 4.5

A subset enumeration tree for $\{1, 2, 3\}$ looks as follows:

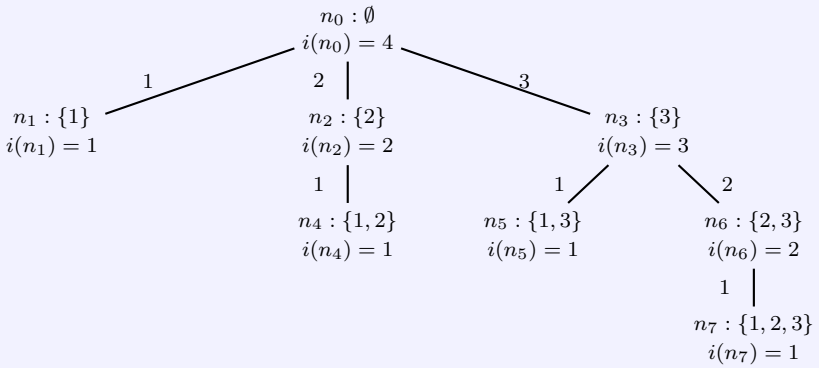


Figure 4.8: Subset Enumeratation Tree.

While in the enumeration tree all elements of $|\mathbb{U}|$ are considered when constructing the child nodes, only a subset is used during the construction with Algorithm HST. In particular, the conflicts in *CS* are ordered and only the first conflict C is used in case the intersection of the conflict and path label h of the vertex v is empty, i.e., $C \cap h(v) = \emptyset$. As in HS-DAG this set is then used as an input to create more child nodes. A global variable *MIN* ensures that each element of $\mathbb{U}$ is mapped to a unique number.

Algorithm 4.5: **HST [Wot01]**

1 Let *MIN* be a variable storing the lowest index not previously assigned to an element of $\mathbb{U}$. Initially, $MIN = |\mathbb{U}|$.

2 Let ci be a bijective function $\mathbb{U} \rightarrow \{1, \ldots, |\mathbb{U}|\}$ that maps each element of $\mathbb{U}$ to an index between 1 and $|\mathbb{U}|$.

3 Let T represent the growing HS-tree. Generate a vertex v which will be the root of the tree. Set $i(v) = |\mathbb{U}| + 1$ and mark v as being opened, i.e., $mark(v) = \square$. The vertex v will be processed in Step 4.

4 Process the nodes in T in a breadth-first order. Nodes are processed in the same order as they are generated. To process an open vertex v do the following:

   a Let $h(v)$ be the set of elements of $\mathbb{U}$ given by the indices i from the root to v, i.e., $h(v) = \{C | ci(C) = i(v')$, where v' is a vertex lying on the path from the root to $v\}$.

   b If for all $x \in CS, x \cap h(v) \neq \emptyset$, then $mark(v) = \checkmark$. Otherwise, label v as open and let y be the first set in CS where $y \cap h(v) = \emptyset$. For every element C from $\mathbb{U}$ in y with no previously defined index ci, let $ci(C)$ be MIN and decrement MIN afterwards. Let $min(v)$ be $MIN + 1$. If $i(v) > min(v)$ create a new array ranging over $min(v), \ldots, i(v) - 1$. Otherwise, let $mark(v) = \times$ and create no child nodes for v.

   c For each n in $min(v), \ldots, i(v) - 1$ create a new vertex v' with $parent(v') = v$, $child(v, n) = v'$, $mark(v') = \square$, and let $i(v')$ be n. The new vertex v' will be processed after all vertices in the same generation as v have been processed.

5 Return the resulting tree T.

By collecting all $h(v)$ of all nodes marked $\checkmark$, we obtain all minimal hitting sets. To ensure an efficient computation and a tree as small as possible, the following rules are applied:

Closing A node v is closed if its associated hitting set is a superset of a hitting set of another previously generated vertex p, i.e.,

$$\exists p \in T : h(p) \subset h(v) \Rightarrow label(v) = \times .$$

Pruning Remove closed nodes v from the tree T. The arc leading from the parent node is removed and the entry in the parents' child array is set to ε. If all entries of the child array are set to ε, the parent node itself is removed from T. Pruning is done until all closed nodes and nodes with only ε entries in *child* are removed from the tree.

Example 4.6

Given a set of conflicts $CS = \{\{a, b\}, \{c, d\}, \{b, c\}, \{d\}\}$, we can construct the universal set $\mathbb{U} = \{a, b, c, d\}$. Figure 4.9 show the constructed tree without the application of the pruning rule, while Table 4.4 records the entire execution of the algorithm. Starting from the root, the first conflict set $\{a, b\}$ is considered and added to the mapping ci. For each element of the conflict a new node is generated. At node n_1 first it is checked whether the path label $h(n_1) = \{b\}$ hits all conflicts in CS. Since it does not, the next element of CS with an empty intersection given the path label is taken, which in this case is $\{c, d\}$. c and d are assigned a unique number by the mapping and for each element a new child node is constructed. n_3 for instance is marked $\checkmark$ since its path label $h(n_3) = \{b, d\}$ hits all conflicts and hence constitutes a minimal hitting set. Also n_9's path label is identified as a minimal hitting set. n_5 is closed due its index, while nodes n_8,n_{10}, and n_{12} are closed since they are supersets of minimal hitting sets already found.

The structure of the tree ensures that not all generated nodes have to be considered during the subset check, but only the nodes marked with $\checkmark$ that are located to the left of the current node with a smaller path length. For instance, consider n_{10} with $h(n_{10}) = \{a, b, d\}$.

To determine whether this node has to be closed due to constituting a superset of an already found minimal hitting set, only the node n_3 has to be considered which is marked ✓ and is in a level above n_{10}. n_9 does not have to be considered since the path length of n_9 is equivalent to the path length of n_{10}.

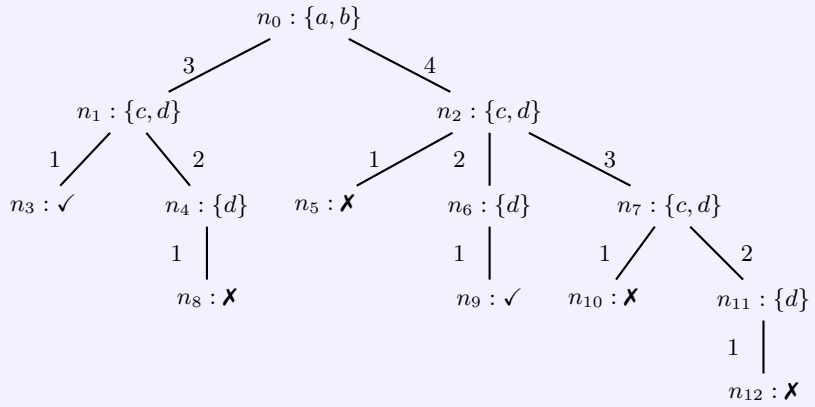


Figure 4.9: Before pruning.

Table 4.4: Execution of HST.

v	$h(v)$	y	***MIN***	$ci(C)$	$min(v)$	$i(v)$	**Action**
n_0	$\emptyset$	$\{a, b\}$	4	$ci(a) = 4$, $ci(b) = 3$	3	5	new $child(n_0, 3) = n_1$
							and $child(n_0, 4) = n_2$
n_1	$\{b\}$	$\{c, d\}$	2	$ci(a) = 4$, $ci(b) = 3$	1	3	new $child(n_1, 1) = n_3$
				$ci(c) = 2$, $ci(d) = 1$			and $child(n_1, 2) = n_4$
n_2	$\{a\}$	$\{c, d\}$	0	$ci(a) = 4$, $ci(b) = 3$	1	4	new $child(n_2, 1) = n_5$
				$ci(c) = 2$, $ci(d) = 1$			and $child(n_2, 2) = n_6$
							and $child(n_2, 3) = n_7$
n_3	$\{b, d\}$		0	$ci(a) = 4$, $ci(b) = 3$			$mark(n_3) = ✓$
				$ci(c) = 2$, $ci(d) = 1$			(hits all C in CS)
n_4	$\{b, c\}$	$\{d\}$	0	$ci(a) = 4$, $ci(b) = 3$	1	2	new $child(n_4, 1) = n_8$
				$ci(c) = 2$, $ci(d) = 1$			
n_5	$\{a, d\}$	$\{b, c\}$	0	$ci(a) = 4$, $ci(b) = 3$	1	1	$mark(n_5) = ✗$
				$ci(c) = 2$, $ci(d) = 1$			($min(n_5) \not< i(n_5)$)
n_6	$\{a, c\}$	$\{d\}$	0	$ci(a) = 4$, $ci(b) = 3$	1	2	new $child(n_6, 1) = n_9$
				$ci(c) = 2$, $ci(d) = 1$			
n_7	$\{a, b\}$	$\{c, d\}$	0	$ci(a) = 4$, $ci(b) = 3$	1	3	new $child(n_7, 1) = n_{10}$
				$ci(c) = 2$, $ci(d) = 1$			and $child(n_7, 2) = n_{11}$
n_8	$\{b, c, d\}$		0	$ci(a) = 4$, $ci(b) = 3$			$mark(n_8) = ✗$
				$ci(c) = 2$, $ci(d) = 1$			(closing)
n_9	$\{a, c, d\}$		0	$ci(a) = 4$, $ci(b) = 3$			$mark(n_9) = ✓$
				$ci(c) = 2$, $ci(d) = 1$			(hits all C in CS)
n_{10}	$\{a, b, d\}$		0	$ci(a) = 4$, $ci(b) = 3$			$mark(n_{10}) = ✗$
				$ci(c) = 2$, $ci(d) = 1$			(closing)
n_{11}	$\{a, b, c\}$	$\{d\}$	0	$ci(a) = 4$, $ci(b) = 3$	1	2	new $child(n_{11}, 1) = n_{12}$
				$ci(c) = 2$, $ci(d) = 1$			
n_{12}	$\{a, b,$		0	$ci(a) = 4$, $ci(b) = 3$			$mark(n_{12}) = ✗$
	$c, d\}$			$ci(c) = 2$, $ci(d) = 1$			(closing)

The pruning rule states that all closed nodes, i.e., nodes marked ✗, may safely be removed. Hence in the example in Figure 4.9, nodes n_5, n_8, n_{10}, and n_{12} are deleted from the tree. Given these removals, we can observe that nodes n_4 and n_{11} have no more children, i.e., all entries of their child array are set to ε. Thus, these nodes are also removed. Subsequently, also n_7 may be deleted from the tree as shown in Figure 4.10.

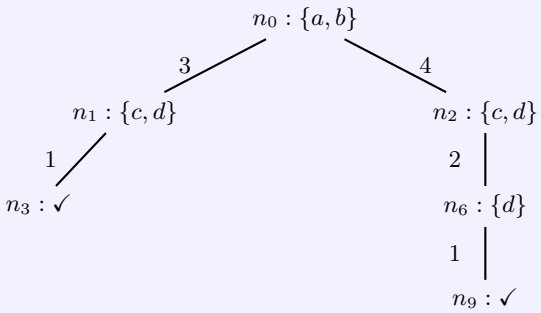


Figure 4.10: After pruning.

Given the resulting tree, we can obtain the minimal hitting sets as the path label of all nodes marked ✓ using ci , i.e., $h(n_3) = \{1, 3\}$ represents the minimal hitting set $\{b, d\}$ and $h(n_9) = \{1, 2, 4\}$ characterizes the minimal hitting set $\{a, c, d\}$.

Binary Hitting Set Tree and the Boolean Algorithm

Lin and Jiang [LJ03] propose the BHS-Tree. First, the tree is constructed by splitting input sets on particular elements and recursively adapting the sets and building the tree. During the bottom up traversal the hitting sets are computed by merging the data of the child nodes.To derive minimal hitting sets a minimization function μ is used to remove any non-minimal solutions.

Definition 4.6 (Binary Hitting Set Tree (BHS-Tree) [LJ03]). Given a minimal set of sets[9] $MCS = C_1, C_2 \ldots, C_n$, a BHS-Tree is a recursive binary-tree, where each node is a tuple (C, H), where C and H are set of sets. The root node is denoted by $(C = MCS, H = \emptyset)$; the left and right children are (C_l, H_l) and (C_r, H_r) respectively. The tree is define recursively as follows:

1. if $C = \emptyset$, then BHS-Tree is an empty tree;
2. else select any element $\alpha \in \bigcup C_i, (C_l = \{C_i \setminus \alpha\} | \alpha \in C_i), H_l = \{\alpha\}$ and $(C_r = \{C_i | \alpha \notin C_i\}, H_r = \emptyset)$.

Thus, while generating the tree, the input of conflicts is decomposed in such a way that the left child handles sets containing the splitting element α, while the right child considers the remaining conflicts. Analogous to Lin and Jiang [LJ03] we denote elements of C by $\langle \cdot \rangle$, while elements of H are contained within $[\cdot]$. Once the tree is constructed it is exploited to derive the minimal hitting sets via Algorithm BHS-Tree. By traversing the tree in a bottom up manner, the hitting sets in H are constructed. In case not all conflicts are known at the beginning of the computation, additional conflicts can be easily accounted for, by adding a new branch to BHS-Tree.

[9] In case the set of conflicts contains any supersets of any C_i, they have to be removed before the construction of the tree.

Algorithm 4.6: **BHS-Tree [LJ03]**

Given a BHS-Tree the minimal hitting sets can be derived with the following rules:

1. If a nodes is a leaf node, then the minimal hitting sets of this node is H; else run Step 2 and 3 recursively
2. Replace every parent node H with $\{H, \{m_l \cup m_r | m_l \in H_l, m_r \in H_r\}\}$. Notice that H may be the empty set.
3. Minimize H at the root node with the function μ until it comprises all minimal hitting sets.

The Boolean algorithm represents the conflict set as a Boolean formula, in such a way that based on Boolean operations the resulting conjunctions correspond to the minimal hitting sets. A set of conflicts *CS* is formalized as a Boolean formula in DNF, where each conflict is represented by a disjunction of negative atoms corresponding to the set's elements. The algorithm recursively applies the function H to the formula to obtain the hitting sets.

Definition 4.7 (Function H(C) [LJ03]). *C* is a Boolean formula, $\neg e$ is the negation of e (both are atoms of the Boolean formula). H is defined recursively:

(1) $H(\bot) = \top$ and $H(\top) = \bot$

(2) $H(\neg e) = e$

(3) $H(\neg e \wedge C) = e \vee H(C)$

(4) $H(\neg e \vee C) = e \wedge H(C)$

(5) $H(C) = e \wedge H(C_1) \vee H(C_2)$, where $\neg e$ is an arbitrary atom of C, $C_1 = \{c | c \in C \wedge \neg e \notin c\}$, and $C_2 = \{c | c \cup \{\neg e\} \in C \vee (c \in C \wedge \neg e \notin c)\}$

Rule **(5)** represents the main strategy behind the method, splitting the search into two portions. One part assumes that the split element e is part of the solution and hence focuses on sets not hit so far. The other part tackles the hitting sets not including e [Pil+11].

Example 4.7

BHS-Tree Given the set $CS = \{\{a, b\}, \{c, d\}, \{b, c\}, \{d\}\}$, first the non-minimal sets, i.e., $\{c, d\}$, are removed. The resulting set $MCS = \{\{a, b\}, \{b, c\}, \{d\}\}$ is used to construct the BHS-Tree in Figure 4.11.

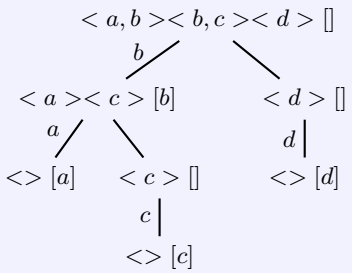


Figure 4.11: BHS-Tree construction

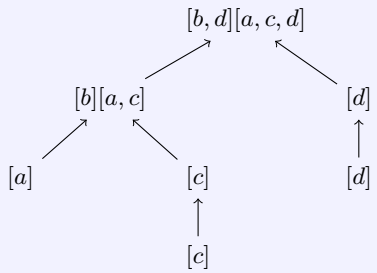


Figure 4.12: Hitting set computation

Figure 4.12 shows the computation of the minimal hitting sets. The minimal hitting sets are generated by traversing the tree in a bottom up manner and updating the information stored in the set H of the visited nodes. Starting at the leafs, each leaf's H remains unchanged. At every other node n, the Cartesian product of the left and right child's H sets

is added to $H(n)$ and the set is minimized with μ removing all supersets. Once at the root node the minimal hitting sets $\{b, d\}$ and $\{a, c, d\}$ can be extracted from H of the root.

Boolean algorithm The Boolean algorithm represents the set *CS*, as a Boolean formula *CSF* in DNF: $CSF = ((\neg a \wedge \neg b) \vee (\neg b \wedge \neg c) \vee (\neg d) \vee (\neg c \wedge \neg d))$. By recursively applying the function H to the Boolean formula, the search space is continuously split and refined.

$$
\begin{array}{lll}
 & H\big((\neg a \wedge \neg b) \vee (\neg b \wedge \neg c) \vee (\neg \mathbf{d}) \vee (\neg c \wedge \neg d)\big) & \textbf{(4)} \text{ with } e = d \\
= & d \wedge H\big((\neg a \wedge \neg \mathbf{b}) \vee (\neg \mathbf{b} \wedge \neg c) \vee (\neg c \wedge \neg d)\big) & \textbf{(5)} \text{ with } e = b \\
= & \big(d \wedge b \wedge H(\neg \mathbf{c} \wedge \neg d)\big) \vee \big(H(\neg a \vee \neg c \vee (\neg c \wedge \neg d))\big) & \textbf{(3)} \text{ with } e = c \\
= & \big(d \wedge b \wedge (c \vee H(\neg \mathbf{d}))\big) \vee \big(H(\neg a \vee \neg c \vee (\neg c \wedge \neg d))\big) & \textbf{(2)} \text{ with } e = d \\
= & \big(d \wedge b \wedge (c \vee d)\big) \vee \big(H(\neg \mathbf{a} \vee \neg c \vee (\neg c \wedge \neg d))\big) & \textbf{(4)} \text{ with } e = a \\
= & \big(d \wedge b \wedge (c \vee d)\big) \vee \big(a \wedge H(\neg \mathbf{c} \vee (\neg c \wedge \neg d))\big) & \textbf{(4)} \text{ with } e = c \\
= & \big(d \wedge b \wedge (c \vee d)\big) \vee \big(a \wedge c \wedge H(\neg \mathbf{c} \wedge \neg d)\big) & \textbf{(3)} \text{ with } e = c \\
= & \big(d \wedge b \wedge (c \vee d)\big) \vee \big(a \wedge c \wedge (c \vee H(\neg \mathbf{d}))\big) & \textbf{(2)} \text{ with } e = d \\
= & \big(d \wedge b \wedge (c \vee d)\big) \vee \big(a \wedge c \wedge (c \vee d)\big) & \text{Distribution and Absorption} \\
= & (b \wedge d) \vee (a \wedge c \wedge d) &
\end{array}
$$

The minimal hitting sets are $\{b, d\}$ and $\{a, c, d\}$.

To remove the unnecessary sub formulas from further consideration, Pill and Quaritsch [PQ12] redefine Rule **(4)** by

(4') $H(\neg e \vee C) = e \wedge H(C_1)$ with $C_1 = \{c | c \in C \wedge \neg e \notin c\}$.

While the original rule does identify e as an element of the current branch, all sets already hit by e are still considered within C. Hence, the improvement ensures that only conjunctions not already hit are considered during computation.

Example 4.7 (cont.)

Assuming $CSF = ((\neg a \wedge \neg b) \vee (\neg b \wedge \neg c) \vee (\neg d) \vee (\neg c \wedge \neg d))$ as before, we apply the function H recursively to the Boolean formula replacing Rule **(4)** with **(4')**:

$$
\begin{array}{lll}
 & H(\neg a \wedge \neg b) \vee (\neg b \wedge \neg c) \vee (\neg \mathbf{d}) \vee (\neg c \wedge \neg d) & \textbf{(4')} \text{ with } e = d \\
= & d \wedge H\big((\neg a \wedge \neg \mathbf{b}) \vee (\neg \mathbf{b} \wedge \neg c)\big) & \textbf{(5)} \text{ with } e = b \\
= & d \wedge \big(b \vee H(\neg \mathbf{a} \vee \neg c)\big) & \textbf{(4')} \text{ with } e = a \\
= & d \wedge \big(b \vee (a \wedge H(\neg \mathbf{c}))\big) & \textbf{(2)} \text{ with } e = c \\
= & d \wedge \big(b \vee (a \wedge c)\big) & \text{Distribution} \\
= & (b \wedge d) \vee (a \wedge c \wedge d) &
\end{array}
$$

Thus in our example, the number of steps necessary to compute the minimal hitting sets $\{b, d\}$ and $\{a, c, d\}$ is reduced.

An essential aspect in regard to the performance of the Boolean algorithm is the heuristic choosing the element e to split in Rule **(5)**. A common idea is to either select an atom e, such that it hits the most sets in *CS*. Pill and Quaritsch [PQ12] improved upon this, by choosing the elements from the smallest C_i in *CS*, to allow processing the smallest conflict set fast.

4.3.2 Empirical Evaluation

In this section, we describe two experiments we have conducted in regard to the most runtime efficient abduction approach for bipartite diagnosis problems. Specifically, we compare

the ATMS and various methods for deriving minimal hitting sets. The first evaluation, is based on two benchmarks; on the one hand, we have taken the same FMEAs we have utilized in the experiments in Chapter 3. On the other hand, we created artificial bipartite diagnosis problems. While the first experiment exploits the Boolean approach, this method lacks the ability to derive hitting sets on the fly. However, in a diagnosis scenario observations may come delayed or additional information may be obtained through probing. In this case, the Boolean approach has to restart the computation from the beginning instead of continuing the process given the new information. Thus, in the second set of experiments we replace the Boolean approach with another method that does possess the capability to resume the hitting set computation given new observations. All results presented below were obtained on a Mac Pro (Late 2013) with a 2.7 GHz 12-Core Intel Xeon ES processor and 64GB of RAM running OS X 10.10.5.

Superiority of the Boolean Algorithm

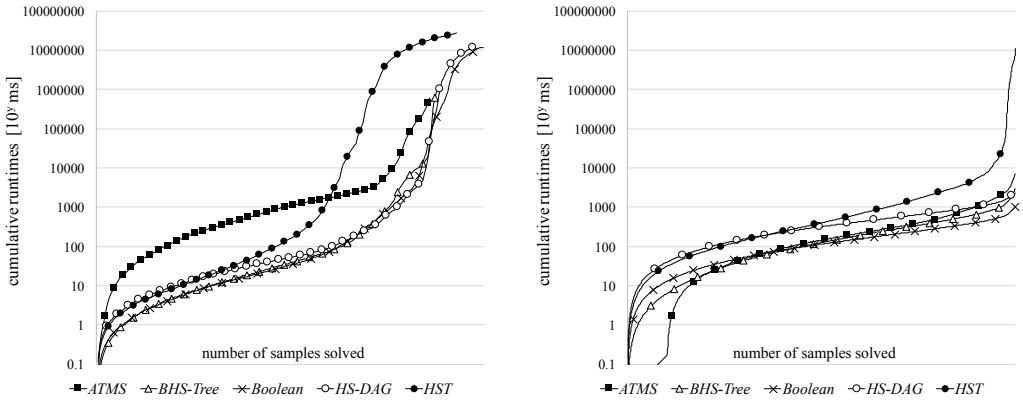
For our first evaluation, we utilized the ATMS as well as the hitting set algorithms described in Subsection 4.3.1. We utilize a Java implementation of Algorithm ABDUCTIVEEXPLANATIONS [Wot+09] based on an *ATMS* [DK86a] as in the previous evaluation. Also the runtime results of *BHS-Tree* [LJ03] stem from a Java implementation from the publicly available diagnosis tool jdiagengine[10] [PW03]. For the remaining hitting set procedures, i.e., *HS-DAG* [Rei87], *HST* [Wot01] and the *Boolean* approach [LJ03], we exploit PyMBD[11] [QP14], a Python library of model-based diagnosis algorithms.

Our data for comparing the abduction methods originate from two separate sources; on the one hand artificial examples and on the other hand a small corpus of FMEAs. In case of the former, we produced examples with a varying number of hypotheses ($10 \leq |Hyp| \leq 500$), effects ($4 \leq |\{A \setminus Hyp\}| \leq 13001$), and clauses ($100 \leq |Th| \leq 13500$). We collected the data on 215 experiments with $|Obs|$ ranging from 1 to 25. The FMEA examples comprises publicly available as well as internally used FMEAs recording fault knowledge from diverse domains[12]. We automatically mapped these failure assessments to abductive knowledge bases, which we can use for logic-based as well as set covering abduction. All in all we conducted experiments on twelve FMEAs with various numbers of hypotheses, effects, and clauses. For each experiment we randomly chose the number of observations as well as the manifestations themselves from the effects available within the model. A total of 2120 experiments where conducted on the FMEA sample set. Clearly, the artificially generated samples are larger in size than the FMEA examples and thus computationally more expensive. Hence, for the evaluation, we enforced a time threshold of 1,000,000 ms, i.e., around 16 minutes.

Figure 4.13 depicts the number of samples solved based on ordered cumulative log runtime. It is apparent that certain methods, such as the *ATMS* or *BHS-Tree*, cannot compute all diagnoses within the given time frame on the artificial examples, while on the practical FMEA samples none of the methods exceeds the threshold. This is also shown in the maximum runtime listed (**MAX**) in Table 4.5; each method's execution is at least one time cut-off at the runtime limit on the synthetic samples. Each sample, where the threshold was exceeded, the computation time was set to 1,000,000 ms for Table 4.5. Given this data, we can conclude that the *Boolean* approach is superior to the other methods, while *HST* preforms poorly on both example sets.

The scatter plots in Figure 4.14 and Figure 4.15 show the direct comparison of two methods. For instance, in Figure 4.14a we see the runtimes of *BHS-Tree* and *ATMS* on the

[10] `www.ist.tugraz.at/modremas/index.html`
[11] `http://modiaforted.ist.tugraz.at/`
[12] The FMEAs contain the additional FMEA characterizing failures of a wind turbine's main bearings.



(a) Artificial benchmark. **(b)** FMEA benchmark.

Figure 4.13: Number of samples solved for growing cumulative log runtime.

Table 4.5: Runtime results for the first evaluation [in ms].

		ATMS	*BHS-Tree*	*Boolean*	*HS-DAG*	*HST*
Artificial Examples	**MIN**	0.05	**0.01**	0.04	0.14	0.11
	MAX	1,000,000.00	1,000,000.00	1,000,000.00	1,000,000.00	1,000,000.00
	AVG	137,811.50	120,449.55	**56,706.66**	76,346.56	353,186.84
	MED	27.14	1.17	**0.89**	1.11	5.92
	SD	341,574.88	321,321.98	**180,694.54**	351,245.37	2,376,405.32
FMEA Examples	**MIN**	< **0.01**	0.01	0.04	0.14	0.09
	MAX	163.81	67.45	**11.12**	23.60	3,652,150.40
	AVG	3.24	1.42	**0.49**	1.12	5,212.47
	MED	0.30	0.27	**0.18**	0.46	0.95
	SD	15.24	6.00	**1.30**	2.61	83,340.27

artificial diagnosis problems. Each data point represents one *PHCAP* computation, where the x-axis value is determined by the runtime of the *ATMS*, while the y-axis value depends on *BHS-Tree*'s execution time. A point located above the diagonal indicates that *ATMS* computed the solution faster for this particular diagnosis problem and a point below symbolizes a superiority of *BHS-Tree* on the example. For the artificial benchmarks, we can further note a dashed line, indicating the runtime limit of 1,000,000 ms. Each point on the dashed line, hence, represents an execution terminated at the runtime threshold. From the scatter plots we deduce that *HS-DAG* and *Boolean* observe a similar runtime behavior, indicated by most data points located closely around the diagonal on both sample sets. However, the *Boolean* method is clearly more efficient. Overall, we discover in the scatter plots that for most comparisons the shape of the data cloud is similar on the FMEA and the artificial examples. An exception is the *ATMS*, which improves on the FMEA samples. Comparing Figure 4.14a to Figure 4.15a, we notice that on the artificially generated samples *ATMS* can outperform *BHS-Tree* only on a few executions. However, this changes on the FMEA diagnosis problems, where the bulk of examples is located above the diagonal in the lower left quadrant or around the diagonal.

Comparing the absolute performance of all algorithms, we observe that the Boolean approach is superior to the other methods. This is due to the efficient strategy for exploring the search space. However, *HS-DAG* and *BHS-Tree* also show good runtime results on both sample sets. For the *ATMS* we can see that while on the simple FMEA benchmarks, the approach is on median acceptable, it is especially slow on the artificial examples, which feature a larger number of hypotheses that have to be considered during the label updates. Also HST could not provide convincing performance results, even though proposed as an optimization of HS-DAG.

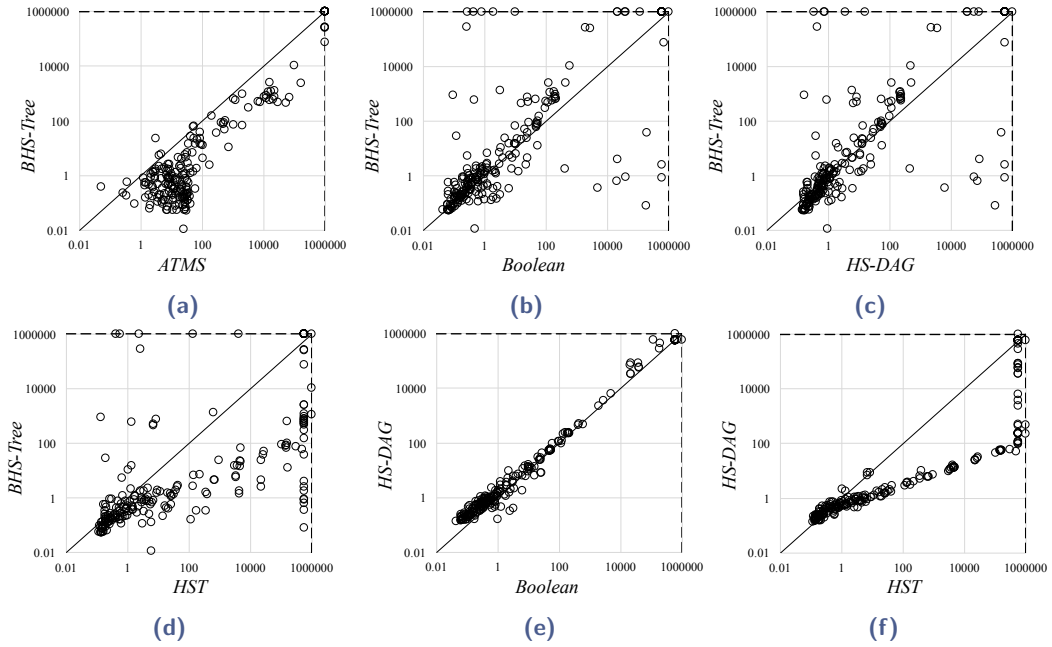


Figure 4.14: Scatter plots comparing the penalized log runtimes on the artificial examples.

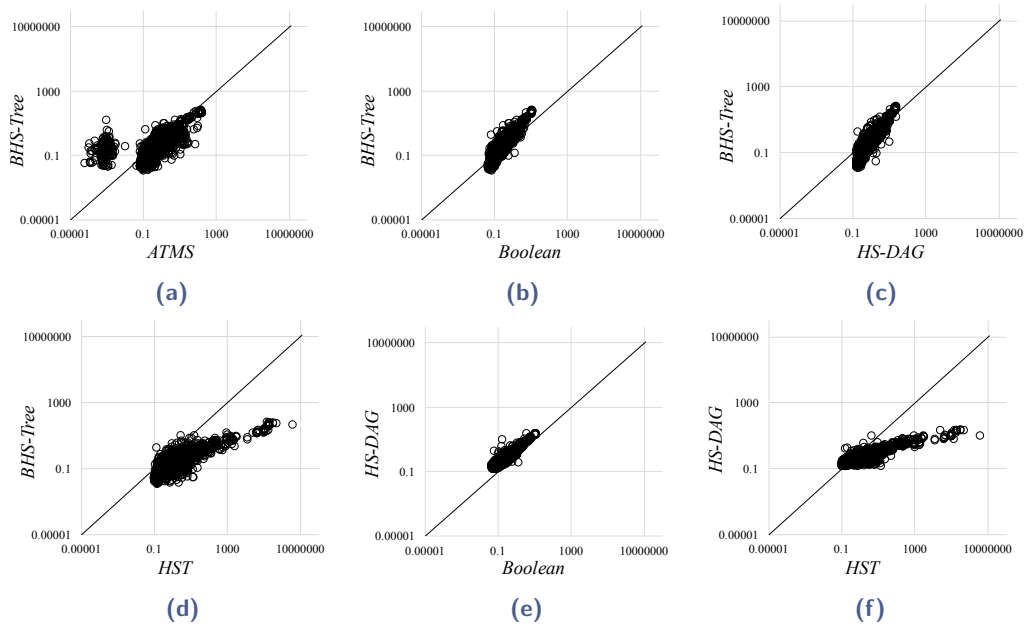


Figure 4.15: Scatter plots comparing the log runtimes on the FMEA examples.

Computing Abductive Diagnoses On The Fly

While the first evaluation conveys the efficiency of the Boolean approach in comparison to the other techniques investigated, the method comes with a disadvantage: all conflicts, i.e., observations, have to be known in advanced. Yet the solutions to a diagnosis problem can be refined, by supplying discriminating observation to the diagnosis engine. Hence in a real world context, the Boolean approach is unfavourable as each additional information would result in a restart of the entire computation. Therefore, we introduce a another algorithm used in the GDE by de Kleer and Williams [DKW87] and is also known as Berge's algorithm [Ber89].

GDE's or Berge's algorithm uses an intuitive notion to compute hitting sets [DKW87; Nyb11]. By definition a hitting set intersected with any set of *CS* is not empty, i.e., each set of *CS* has to contribute to each hitting set. In Berge's algorithm the minimal hitting sets are constructed and modified incrementally. Initially, the set of minimal hitting sets *MHS* is empty. Whenever a new $C \in CS$ is to consider, each $H_i \in MHS$ is checked whether $C \cap H_i = \emptyset$. In case it is not, i.e., H_i already hits C, H_i remains unchanged, otherwise it is removed from *MHS* and for each $c \in C$ a new set is created containing H_i and c. By checking whether subsets are present within *MHS* the algorithm ensures to derive minimal hitting sets.

Algorithm 4.7: BERGE'S ALGORITHM [Nyb11]

Require: *CS*: a set of sets
Ensure: *MHS*: set containing all minimal hitting sets of *CS*

```
1:  MHS ← ∅
2:  for all C ∈ CS do
3:      MHS_old ← MHS
4:      MHS_add ← ∅
5:      for all H_i ∈ MHS do
6:          if H_i ∩ C = ∅ then
7:              MHS_old ← MHS_old \ H_i
8:              for all c ∈ C do
9:                  H_new ← H_i ∪ {c}
10:                 for all H_k ∈ MHS, H_k ≠ H_i do
11:                     if H_k ⊆ H_new then
12:                         goto L1
13:                     end if
14:                 end for
15:                 MHS_add ← MHS_add ∪ {H_new}
16:                 L1
17:             end for
18:         end if
19:     end for
20:     MHS ← MHS_old ∪ MHS_add
21: end for
22: return MHS
```

Example 4.8

We consider the same set $CS = \{\{a, b\}, \{c, d\}, \{b, c\}, \{d\}\}$ as in Subsection 4.3.1. Assume that the algorithm has already processed the sets $\{a, b\}, \{c, d\}, \{b, c\}$. Hence, *MHS* currently consists of $\{a, c\}, \{b, c\}, \{b, d\}$. We start in Line 2, where the new conflict to consider from *CS* is $\{d\}$.

In Line 5, the first H_i from *MHS* is $\{a, c\}$; $\{a, c\} \cap \{d\} = \emptyset$ hence the if-branch is executed and MHS_{old} then comprises $\{b, c\}, \{b, d\}$. In Line 9 the new hitting set H_{new} is built as $\{a, c\} \cup \{d\}$. Since there is no hitting set in $MHS \setminus \{a, c\}$ which is a subset of $\{a, c, d\}$ it is added to MHS_{add}.

Back in Line 5, the next hitting set $\{b, c\}$ is considered, which leads to $H_{new} = \{b, c, d\}$. Yet, since there is a subset of H_{new} already contained in *MHS* ($\{b, d\} \subseteq \{b, c, d\}$) the condition in Line 11 evaluates to true, hence the execution jumps to label `L1`. The last set

of *MHS* to check is $\{b, d\}$. Since $\{b, d\} \cap \{d\} \neq \emptyset$ and all sets in *MHS* have been checked, in Line 20 *MHS* is constructed as $MHS = \{\{b, d\}, \{a, c, d\}\}$. As all sets in *CS* have been processed *MHS* is returned.

In the second evaluation, we replace the *Boolean* approach implementation of PyMBD with a Java version of *Berge*'s algorithm. Further, the second evaluation focuses on artificial examples. We obtained new synthetic samples with different numbers of hypotheses, effects, and connections. A total of 195 samples were created with a varying number of hypotheses ($12 \leq |Hyp| \leq 3120$), effects ($1 \leq |\{A \setminus Hyp\}| \leq 5000$), clauses ($12 \leq |Th| \leq 5100$), and observations ($1 \leq |Obs| \leq 30$).

Table 4.6: Runtime results for the second evaluation [in ms].

		ATMS	*Berge*	*BHS-Tree*	*HS-DAG*	*HST*
Artificial Examples	**MIN**	0.22	**0.03**	0.06	0.20	0.13
	MAX	5,444.11	375.85	435.96	**371.37**	592,876.17
	AVG	69.05	10.15	10.92	**6.12**	5,570.71
	MED	1.83	**0.51**	**0.51**	0.88	0.95
	SD	520.54	44.14	49.10	**35.41**	56,273.79

From Table 4.6 and Figure 4.16 we conclude that in contrast to the first evaluation there is no clearly superior algorithm. In comparison, *HS-DAG* positions itself as the most efficient approach on average, while *BHS-Tree* and *Berge* provide good runtime results on median. From Figure 4.17b, we conclude that *Berge* and *BHS-Tree* have a very similar runtime behavior indicated by the data points accumulating on the diagonal. Overall the results are in line with the one's we have seen in the previous evaluation, with the additional results on *Berge* which provides good results on the test set.

4.4 Conclusion

In this chapter, we have focused on the computation of abductive diagnosis for problem descriptions stemming from FMEAs. The first investigation deals with conflict driven techniques to derive diagnoses, i.e., proof-tree style abduction. Here, we augment the theory with the negated observation and obtain explanations as refutations of the resulting formula. We reviewed four different algorithms to compute abductive explanations for a propositional diagnosis problem. We compared the direct strategy based on an ATMS to three approaches relying on conflict computation of an unsatisfiable model. The first two methods are based on computing conflict sets in the context of infeasibility analysis, i.e., MUSes. These unsatisfiable cores constitute the minimal abductive explanations. Since the computation of MUSes is computationally demanding we proposed two set-ups. In the direct version, we utilize an MUS enumeration tool, while in the indirect variant we exploit its hitting set dual, MCSes, in order to derive minimal diagnoses. The last method utilizes HS-DAG in combination with a theorem prover to derive refutations on demand, which in our case constitute the explanations.

In our tests, the conflict-based methods did not offer advantages against the ATMS. The SAT-based approaches have the drawback of not being focused on the set of abducibles, but rather enumerate all sets regardless of the clause corresponds to a hypothesis or not. Further, we could observe that in fact MCS enumeration and subsequent hitting set computation is preferable to the direct MUS approach. Surprisingly, HS-DAG did not perform well even on the smaller examples. We explain this, by the encoding of the problem, which has not been

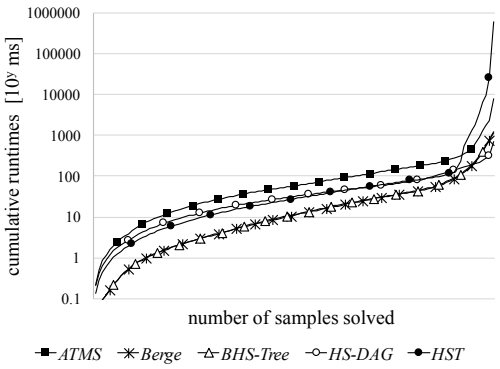


Figure 4.16: Cumulative runtimes on the second artificial benchmark.

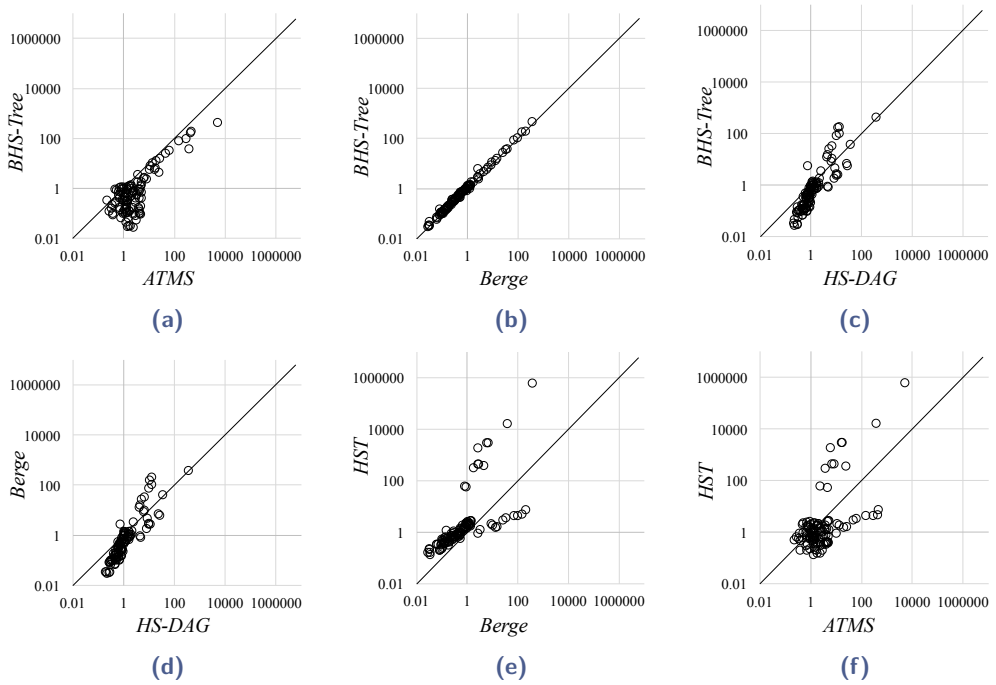


Figure 4.17: Scatter plots comparing the log runtimes on the second set of artificial examples.

ideal for utilized theorem prover. Despite the fact that the data provided no evidence of a computational benefit in employing a SAT-based approach, we believe that the possibility to utilize more expressive models provides an interesting incentive.

Based on the structural characteristics of the models, we examine Thus, we can take advantage of the simplicity of the FMEA models, i.e., they contain bijunctive definite Horn clauses where a single assumption implies a single effect, when computing explanations. In particular, we exploit the notion of the simple set-covering problem, which can be reframed as a hitting set problem. We discussed Greiner et al.'s HS-DAG [Gre+89], Wotawa's HST [Wot01] and BHS-Tree as well as the Boolean algorithm [LJ03]. In the first empirical experiments we could see a clear superiority of the Boolean approach, which has also been discovered in the comparison conducted by Pill, Quaritisch, and Wotawa [Pil+11], where for the consistency-based case the Boolean method was the most efficient contender on unrestricted searches. Further, we could see in our evaluation similar to theirs that HS-DAG operates well in case of large conflict sets, i.e., in our case a large number of observations.

Unfortunately, the Boolean algorithm requires that in order to compute all hitting sets all conflicts characterizing the problem have to be known in advance. Considering the process we have depicted in the previous chapter, troubleshooting benefits from additional observations in order to refine the diagnosis results meaning the set of observations to consider can adapt on the fly. Thus, in the second evaluation we have replaced the Boolean approach with Berge's algorithm that given a set of hitting sets and a new—not necessarily minimal—conflict derives an updated set of minimal hitting sets. Our evaluation could not confirm the results of Pill, Quaritisch, and Wotawa [Pil+11] that Berge's algorithm is on several samples on par with the Boolean approach. In particular, we could see that while Berge's algorithm performed similar to BHS-Tree, the HS-DAG has shown most promising on our sample set. Especially from the scatter plots of the artificial sample set, we observe that Berge/BHS-Tree provide superior runtime results on different samples than HS-DAG indicating that certain instances are "simpler" to Berge/BHS-Tree than HS-DAG and vice versa.

5 Faster Horn Diagnosis - Finding Explanations in Horn Clause Abduction

> *There should be no combination of events for which the wit of man cannot conceive an explanation.*
>
> — **Sir Arthur Conan Doyle**
> "The Valley of Fear". 1915

The content of this chapter has not been published yet.

5.1 Motivation

While in the previous chapter, we have focused on the simple models obtained from FMEAs, we want to direct our attention to a more general case, i.e., Horn clauses. Traditionally, abductive model-based diagnosis exploits a representation of the faulty system behavior that can often be straightforwardly phrased in Horn clause theories [CT91]. A formalization using this subset of logic has the advantage that diagnosis is computationally less complex than on general theories [EG95].

Example 5.1

In the following example, we describe a simple diagnosis problem taken from the industrial wind turbine domain. Gearbox lubrication is an essential aspect in regard to the reliability of wind turbines as it protects the contact surfaces of gears and bearings from excessive wear and prevents overheating. In a simplified scenario, the insufficient lubrication can be caused by various component failures. On the one hand, a broken oil pump leads to reduced oil pressure. This decreased pressure diminishes the flow rate of oil through the system. On the other hand, damages to the oil cooler eventually cause an overheating of the oil, which in conjunction with a blocked oil filter also negatively affects the lubrication resulting from a reduction in the film thickness at bearing and gear contacts. These relations can be described by a set of Horn clauses $HC = \{Damaged\_pump \rightarrow reduced\_pressure, reduced\_pressure \rightarrow poor\_lubrication, Broken\_filter \wedge overheating \rightarrow poor\_lubrication, Cooler\_leaks \wedge Cooler\_cracks \rightarrow overheating\}$. Given the interactions between component faults and their consequences, we can identify two root causes of insufficient lubrication: (1) a damaged oil pump and (2) a broken filter in conjunction with cracks and leaks of the oil cooling component. These causes constitute the faults we want to identify during diagnosis. In the remainder of this paper, we make use of this running example to illustrate the different abductive reasoning approaches.

As abductive inference can be applied to different tasks, various techniques for solving logic-based abduction problems have emerged. We have seen the strategy of deriving explanations as the search for logical consequences, i.e., diagnoses are derived deductively. For

instance, we have discussed kernel resolution [SDV01] that restricts the computed consequences to clauses of interest. This technique can be implemented efficiently with ZBDDs and performs well on large problem instances. Inoue [Ino92] proposes Skipping Ordered Linear tableau calculus (SOL-tableau calculus) for extracting consequences of interest equivalent to abductive explanations. Another classical technique regarding abduction is Abductive Logic Programming (ALP) [Kak+92] which has been introduced as a means to declaratively solve abduction problems within the framework of logic programming. Logic programming is augmented with abducible predicates, i.e., ground instances that may be part of a diagnosis, and integrity constraints posing restrictions on what constitutes an admissible solution. Besides specific tools such as the $\mathcal{A}$-System [Kak+01], ALP can be realized using Answer Set Programming (ASP) [Sch16]. In the previous section, we have seen proof-tree completion-style abduction. Here the diagnosis problem is rewritten in an entailment preserving way, such that explanations are equivalent to conflicts [McI98]. Using this notion of abduction, innovations in extracting MUSes of logical formulae, which are equivalent to conflicts, can be exploited to obtain abductive diagnoses [Lif+16]. Recently, inspired by the success of implicit hitting set algorithms[1] for MaxSAT, Saikko, Wallner, and Järvisalo [Sai+16] have introduced a method that computes one minimum-cost abductive explanation by utilizing Integer Programming to iteratively derive hitting sets. Each hitting set is then checked whether it constitutes a solution. Ignatiev, Morgado, and Marques-Silva [Ign+16] report on an improvement of this method by deriving the hitting sets using a MaxSAT solver.

Yet contemplating the collection of approaches and tools for abduction, the question, which strategy to follow, remains. An analysis seeking to answer this question for consistency-based diagnosis algorithms exists [Nic+13]. The authors compare two strategies to derive diagnoses: on the one hand methods that directly compute the solutions given the system description and symptoms and on the other hand conflict-based techniques that exploit contradictions arising from correct behavior assumptions and the observations. Another study aiming at comparing different fault identification techniques by Feldman et al. [Fel+10a] introduces a benchmark framework for executing diagnosis methods for physical systems under identical conditions. The performance evaluation presented includes rule-based, (consistency) model-based, data driven, and stochastic fault detection and identification approaches.

While there is research comparing the consensus among different abduction formalizations [McI98], assessing an emerging tool's performance in regard to previously proposed mechanisms [Sai+16; Egl+00], or applying a reasoning system to different domains [NM92], a broader empirical evaluation as available in the consistency-based case is—to the best of our knowledge—missing for abductive Horn diagnosis. For instance, McIlraith [McI98] reviews different formal characterizations, frameworks, techniques, and application areas for abductive inference. While the comparison is broad and extensive, it is only theoretic in nature. Egly et al. [Egl+00] describe a translation of different non-monotonic reasoning tasks, such as abduction, into the evaluation problem of quantified boolean formulas and compare an implementation of their method to non-monotonic reasoning solvers, such as dlv [Eit+98] or Theorist [Poo+87], on multiple benchmark problems. The focus of this work is to show whether non-monotonic reasoning tasks can be solved efficiently by translation to quantified boolean formulas; hence, only a portion of the assessment concerns abduction. A similar undertaking can be reported in the context of consequence finding, where Simon and del Val [SDV01] examine the capabilities of kernel resolution on structured and random instances for prime implicate, knowledge compilation, abduction, and consistency-based diagnosis. Ng and Mooney [NM92] have conducted an empirical analysis comparing the

[1] Implicit hitting set algorithms iteratively exclude answers that are not solutions based on hitting sets over these non-solutions.

general abduction system ACCEL to a set of problems in plan recognition as well as consistency model-based and set-covering diagnosis. While the authors could show that a general solver can derive explanations within the two application domains in reasonable time, their experiments consider solely a single tool.

Hence in this chapter, we aim at providing guidance for deciding on an abduction procedure focusing on propositional Horn theories. Our work is based on the previous chapter's research, in which we have assessed abductive inference on models restricted to bijunctive Horn clauses. Besides adapting and extending the portfolio of algorithms investigated, we further consider more expressive Horn representations in this chapter. In particular, we compare the performance of propositional Horn clause abduction within two general directions, i.e., direct and conflict-driven techniques, similar to the work by Nica et al. [Nic+13] in the consistency-based case. The direct abduction methods include the ATMS and a tool for consequence finding, while for the proof-tree style abduction methods we improve upon two techniques we have already seen in the previous chapter and utilize an encoding of abductive reasoning to exploit an ASP solver. The improvements were applied on the one hand to the refutation search with the HS-DAG, were we employ a minimization technique to keep the graph small and on the other hand to the direct MUS computation that we have now framed in an assumption-based manner to fit our goal of deriving abductive explanations. To identify runtime trends, we conducted an empirical evaluation in Section 5.3 based on publicly available tools, implementations of various traditional abductive reasoning algorithms, and adaptations of recent and proven techniques. The diagnosis problems utilized within our analysis are taken on the one hand from real-world domains and on the other hand are artificially created similar to practical fault identification scenarios. One of our goals is to discover whether a certain approach is dominant on these reasonable samples. Besides seeking information on efficiency differences between direct proof and conflict-driven algorithms, we aim at showing to what extend more general off-the-shelf tools can compete with specific Horn reasoning methods.

5.2 Selected Diagnosis Algorithms

In this section, we describe our selected procedures comprising direct reasoning methods and conflict-driven approaches. Direct proof procedures derive explanations via logical consequences, while conflict-driven techniques generate refutations essentially constituting abductive diagnoses. On the one hand, we have chosen methods, such as the ATMS, consequence finding with linear resolution, and conflict driven search via the HS-DAG, which have been applied frequently in consistency-based diagnosis and abduction. On the other hand, we want to contrast those well-known techniques to more recent mechanisms. Particularly, we take advantage of ASP as well as developments in infeasibility analysis.

All in all we present five general abduction methods as well as novel adaptations of some of the techniques aiming at improving efficiency. For each algorithm an overview of its functionality and how to use it in the context of abductive model-based diagnosis is provided. Note here that depending on the utilized implementations and tools the encoding may differ from the one we propose. The first two set-ups, i.e., the ATMS and abduction via consequence finding, can be attributed to the direct proof methods, while the remaining techniques all derive explanations through refutations. Since we intend to assess the performance off-the-shelf reasoning engines can achieve, we propose two publicly available tools that can be used for abductive diagnosis.

5.2.1 Abduction with the ATMS

We have already discussed the workings of the ATMS in Section 2.3.1. Here, we just want to show again the operation of the ATMS on the chapter's running example.

Example 5.1 (cont.)

Figure 5.1 depicts the And-or-graph internally constructed by the ATMS based on the first three clauses mentioned in the theory, where the label Λ is shown in red underneath each node's name. The label of the node *overheating* is empty because there is no knowledge available regarding its truth state, while for the remaining nodes the label values are determined by themselves and their antecedent nodes. Figure 5.2 shows the graph after the information of the last Horn clause is propagated through the data structure. The label of the node corresponding to the observation contains the parsimonious abductive diagnoses, $\Lambda(poor\_lubrication) = \Delta$-Set.

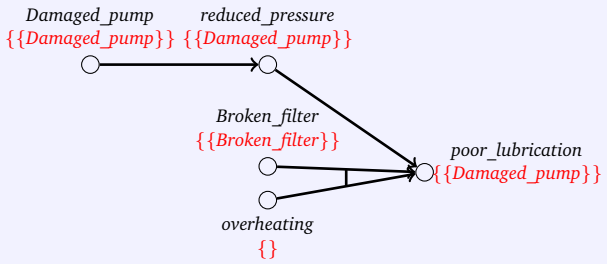


Figure 5.1: And-or-graph considering the first three clauses of *Th*.

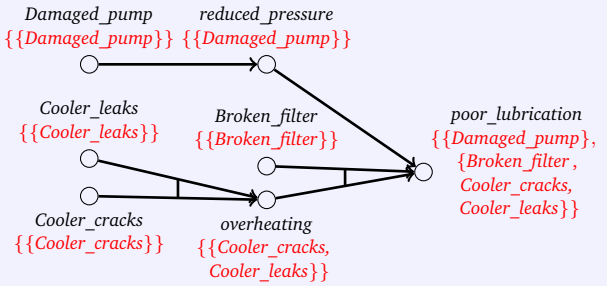


Figure 5.2: And-or-graph after propagating all Horn clauses of *Th*.

5.2.2 Abduction as Consequence Finding via SOL-resolution

Consequence finding provides a general framework for solving a variety of reasoning problems by obtaining logical consequences from a knowledge base given a production field [Mar00]. Recall a production field $\mathcal{P}\langle\mathfrak{L}_\mathcal{P}, Cond\rangle$ restricts the generated clauses to a subset of the representation language that is of interest in the given context. These limitations can be enforced upon the form of literals found in the derived consequences ($\mathfrak{L}_\mathcal{P}$), e.g., only containing abducibles, and on general conditions of the solution (*Cond*), e.g., consequences with a cardinality of 1 [Ino92].

Inoue has shown a procedural interpretation of ATMS abduction based on Skipping Ordered Linear Resolution (SOL-resolution) [Ino91] and has mechanized consequence finding via SOL-tableau calculus for first-order theories [Ino92]. SOL-resolution is an extension of linear resolution with ordered clauses augmented with an additional skip

rule allowing to bypass literals belonging to $\mathcal{P}$ instead of resolving them. It is sound and complete in regard to deriving consequences, which are also referred to as characteristic clauses. Characteristic clauses fulfill the requirements of the production field and are minimal with respect to subsumption.

***Algorithm 5.1:* SKIPPING ORDERED LINEAR DEDUCTION (SOL-DEDUCTION) [Ino91]**

Given a production field $\mathcal{P}$, an SOL-deduction of a clause S from a theory $Th \cup Top$, where *Top* is the top clause of the deduction, is a sequence of structured clauses $D_0 \dots D_n$. A structured clause is a pair $\langle P, Q \rangle$ consisting of a clause P and an ordered clause Q. Q may contain framed literals, which are literals already resolved upon. Structured clauses are generated iteratively starting with $D_0 = \langle \emptyset, Top \rangle$ and ending with $D_n = \langle S, \emptyset \rangle$. Each proceeding clause $D_{i+1} = \langle P_{i+1}, Q_{i+1} \rangle$ is constructed from $D_i = \langle P_i, Q_i \rangle$ by the following rules [Ino91]:

a. Let l be the left most literal of Q_i:
   i. (*skip*) If $P_i \cup l$ belongs to $\mathcal{P}$, then l is skipped, i.e., $P_{i+1} = P_i \cup l$ and Q_{i+1} is the ordered clause obtained by removing l from Q_i .
   ii. (*resolve*) If there is a clause B_i in *Th* such that l can be resolved with literal $k \in B_i$, then B_i and Q_i are concatenated in Q_{i+1}, such that k is removed and l is framed.

b. (*reduce*) Delete any unframed literal k in Q_{i+1} for which there exists a framed literal $\neg k$. Further, merge right any identical literals in Q_{i+1}.

c. (*truncation*) Every framed literal not preceded by an unframed literal is deleted from Q_{i+1}.

Skip and *Resolve* are not exclusive operations. In addition, $P_i \cup Q_i$ may not be a tautology and may not be subsumed by any $P_j \cup Q_j$, where $\langle P_j, Q_j \rangle$ is a previously constructed structured clause.

Abduction is a significant application of consequence finding. Recall that we can rewrite the relation of diagnoses, theory, and observations so that a diagnosis is a logical consequence. In this case, the negation of an explanation can be seen as an interesting theorem which is obtained as the characteristic clause from the theory and the negation of the observation to be explained. Algorithm CONSQXPLAIN describes a procedure to compute abductive diagnoses for a *PHCAP* under consequence finding semantics assuming a method for enumerating consequences, such as SOL-resolution, is available. To extract explanations the background theory is enriched with the negation of the symptoms to be explained, i.e., the clause $\bigvee_{o_i \in Obs} \neg o_i$ is added as a top clause. Further, only clauses consisting of abducibles are of interest; thus, the production field comprises the negated hypotheses, i.e., $\forall h \in Hyp : \neg h \in \mathfrak{L}_{\mathcal{P}}$. As no other conditions should be imposed upon the generated characteristic clauses, we do not define *Cond*. Using a consequence finding enumeration procedure (`EnumConsq`), all characteristic clauses solely consisting of negated hypothesis propositions are obtained. Lastly, the hypotheses need to be transformed to their positive counterpart to be equivalent to the abductive diagnoses[2].

[2]It is important to note here, that in contrast our Definition 2.22 assumes a consequence consists of the positive assumptions and not their negations. Thus, Definition 2.22 already implicitly assumes a transformation from the consequences to their positive equivalents.

Algorithm 5.2: CONSQXPLAIN

Require: A: the set of propositional variables, Hyp: the set of hypotheses, Th: the Horn theory, Obs: the set of observations
Ensure: Δ-Set: set of minimal diagnoses

1: $\Delta\text{-Set} \leftarrow \emptyset$
2: $Top \leftarrow \bigvee_{o \in Obs} \neg o$ ▷ Top clause consisting of negated observations
3: $\mathcal{P} \leftarrow \langle \bigcup_{h \in Hyp} \neg h \rangle$ ▷ Specification of the production field
4: $\mathcal{CC} \leftarrow \texttt{EnumConsq}(\mathcal{P}, Th, Top)$ ▷ Characteristic clause enumeration
5: **for** $\mathcal{C} \in \mathcal{CC}$ **do**
6: $\quad \Delta\text{-Set} \leftarrow \Delta\text{-Set} \cup \bigcup_{\neg h \in \mathcal{C}} h$ ▷ Transformation of hypothesis propositions to their positive equivalent
7: **end for**
8: **return** Δ-Set

Example 5.1 (cont.)

Considering our lubrication example from the previous section, we can reformulate the abduction problem for consequence finding. For clarity, we repeat the theory's Horn clauses this time as disjunctions:

$$Th = \left\{ \begin{array}{c} (1)\ \neg Damaged\_pump \vee reduced\_pressure, \\ (2)\ \neg reduced\_pressure \vee poor\_lubrication, \\ (3)\ \neg Broken\_filter \vee \neg overheating \vee poor\_lubrication, \\ (4)\ \neg Cooler\_leaks \vee \neg Cooler\_cracks \vee overheating \end{array} \right\}.$$

The top clause contains the negation of the observations, i.e., $Top = \neg poor\_lubrication$, and the production field comprises the negations of the abducible hypotheses, i.e., $\mathcal{P} = \langle \{\neg Damaged\_pump, \neg Broken\_filter, \neg Cooler\_leaks, \neg Cooler\_cracks\} \rangle$. We can derive the characteristic clause $\neg Damaged\_pump$ from $Th \cup Top$ and $\mathcal{P}$ by SOL-deduction as follows:

1. $\langle \emptyset, \langle \underline{\neg poor\_lubrication} \rangle \rangle$ — *top* clause
2. $\langle \emptyset, \langle \underline{\neg reduced\_pressure}, \boxed{\neg poor\_lubrication} \rangle \rangle$ — resolve $\neg poor\_lubrication$ with (2)
3. $\langle \emptyset, \langle \underline{\neg Damaged\_pump}, \boxed{\neg reduced\_pressure}, \boxed{\neg poor\_lubrication} \rangle \rangle$ — resolve $\neg reduced\_pressure$ with (1)
4. $\langle \neg Damaged\_pump, \emptyset \rangle$ — skip $\neg Damaged\_pump$ and truncation of $\neg reduced\_pressure$ and $\neg poor\_lubrication$

Given Definition 2.22 and 2.23 of a consequence and a consequence-finding diagnosis, *Damaged_pump* constitutes a minimal solution. Similarly, the second explanation can be obtained.

While SOL-resolution was proposed in a chain format similar to model-elimination, another technique developed uses a framework of connection tableau calculus [Nab+10]. This method bears advantages such as allowing to retain already solved parts. A tableau $\mathcal{T}$ is a

labeled ordered tree, where every node except the root node is labeled by a literal. Leaf nodes can further be marked as **closed** or **skipped**. If all leafs are marked, a tableau is considered solved. A selection function is used for assigning a subgoal to every unsolved tableau based on its unmarked leafs. SOL-resolution in this framework consists of a sequence of tableaux, where each tableau $\mathcal{T}_{i+1}$ is constructed from $\mathcal{T}_i$ by first selecting a subgoal to pursuit and then applying a set of inference rules (*skip*, *extension*, *factoring*, and *reduction*). These rules are based on the inference rules for SOL-resolution described above. To derive all consequences a procedure has to visit all nodes in a SOL-search tree, which explicitly expresses all possible SOL-deductions, and enumerate the consequences of each tableau.

Example 5.1 (cont.)

Considering Example 5.1. We show how to derive the diagnosis *Damaged_pump* based on tableau calculus. First, the top clause is added below the root node (Figure 5.3). Since the only leaf ¬*poor_lubrication* is unmarked, it represents the only possible subgoal. Given the leaf and clause (2) we can resolve, i.e., extend the tableau with ¬*reduced_pressure* ∨ *poor_lubrication* (Figure 5.4). Since ¬*poor_lubrication* was resolved with *poor_lubrication*, we mark the corresponding node **closed**. The only new subgoal is ¬*reduced_pressure* and again we extend the tableau this time with clause (1). The remaining subgoal is ¬*Damaged_pump* which belongs to $\mathcal{P}$; thus, the node is marked as **skipped** and since all leafs are marked the tableau in Figure 5.5 is solved. The derived characteristic clause is ¬*Damaged_pump*, which correspond to the diagnosis *Damaged_pump*.

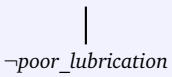


Figure 5.3: Depth 1.

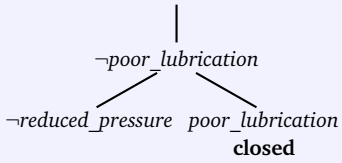


Figure 5.4: Depth 2.

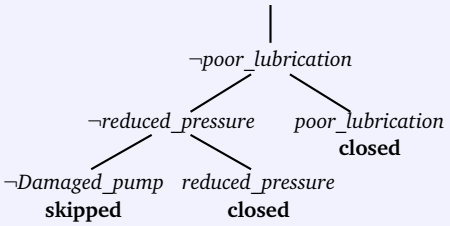


Figure 5.5: Depth 3.

5.2.3 Conflict-Driven Search via HS-DAG

In consistency-based diagnosis, root causes are identified by detecting conflicts, which are essentially assumptions contradicting the encountered manifestations. Within the consistency-based framework, contradictions and diagnoses exhibit a hitting set relation, i.e., by determining all conflicts and subsequently computing their minimal hitting sets all parsimonious diagnoses can be extracted [Rei87; DKW87].

Reiter [Rei87] proposes an approach to derive parsimonious consistency-based diagnoses from conflicts by maintaining a tree structure and exploiting a theorem prover to infer refutations on demand. Some inadequacies of the original algorithm in regard to non-minimal conflict sets were corrected by Greiner et al. [Gre+89] and they devised the method HS-DAG which we described in Subsection 4.3.1. As we have seen in previously to utilize the

conflict-driven search via HS-DAG for abductive diagnosis, a theorem prover *TP* is required to extract refutations during the construction of the graph. As stated by Reiter, the algorithm is relatively independent of the underlying logic and prover; thus, special purpose theorem provers can be utilized to complement the diagnosis domain. Since we are focusing on Horn theories, LTUR [Min88] is a suitable candidate. Algorithm HS-DAGXPLAIN[3] describes a simplified version of the basic approach of HS-DAG and how to exploit it for abductive diagnosis. Within the proof tree type-abduction we are proposing, we need to create a contradicting theory and supply it to the theorem prover. The conflict search and diagnosis computation starts with a root node. The root has an empty edge label and represents at this stage the last level of the graph, i.e., *lastlevelnodes*. As long as there are open nodes to process in the last level, nodes are used for deriving new conflicts or minimal hitting set. To determine whether a node is a minimal hitting set, a consistency check is necessary assuming all hypotheses except the ones in the edge label of the current node are true. We use the method `checkConsistency`(T,F), where T as well as F is a set of assumptions. `checkConsistency` propagates the value *true* (*false*) for all assumptions in T (F) and subsequently returns the contradicting assumptions or the empty set in case the theorem prover is consistent. In case it is consistent, the current node is a minimal hitting set and marked ✓. Otherwise, the conflict computed by *TP* is added to the set of conflicts *CS*. For each element in the conflict a new node is created with a new edge label consisting of the conflict element and the parent's edge label. The newly created nodes are added to the *lastlevelnodes* to be considered in the computation. Once the entire graph has been processed, i.e., there are no more open nodes, all conflicts have been computed. Subsequently, they have to be tested for consistency with the original theory using *TP* to ensure consistent explanations. Note here that the diagnoses returned may not be parsimonious depending on the theorem prover.

Algorithm 5.3: **HS-DAGXPLAIN (based on [Gre+89])**

Require: *A*: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: the Horn theory, *Obs*: the set of observations
Ensure: Δ-Set: set of diagnoses
  $CS \leftarrow \emptyset$
  Δ-Set $\leftarrow \emptyset$
  $TP \leftarrow Th \wedge (\bigwedge_{o \in Obs} o \rightarrow \bot)$ ▷ Create contradictions
  $root \leftarrow$ **new** `Node`$(\emptyset)$ ▷ New root node with an empty edge label
  $lastlevelnodes \leftarrow < root >$
  while $lastlevelnodes \neq \emptyset$ **do** ▷ Process last level
    $newnodes \leftarrow \emptyset$
    for all $n \in lastlevelnodes$ **do**
      $\mathcal{CO} \leftarrow$ `checkConsistency`$(Hyp \setminus h(n), h(n))$ ▷ Check consistency and return contradiction
      if $\mathcal{CO} \neq \emptyset$ **then**
        $CS \leftarrow CS \cup \mathcal{CO}$ ▷ *TP* is inconsistent; add the new conflict
        for all $c \in \mathcal{CO}$ **do**
          $newnodes \leftarrow newnodes \cup$ **new** `Node`$(c \cup h(n))$ ▷ Create new node with new edge label
        end for
      else

[3]For simplicity we have not included the pruning enhancements proposed by Greiner et al. [Gre+89] in the algorithm description even though these refinements may be necessary depending on whether the theorem prover ensures to generate non-minimal refutations or not. Subsection 4.3.1 contains a more elaborate description of the algorithm.

```
            n.mark ← ✓                                  ▷ TP is consistent; minimal HS
        end if
        lastlevelnodes ← lastlevelnodes ∪ newnodes
    end for
end while
return IGNOREINCONSISTENT(Hyp,Th,CS,Δ-Set)

function IGNOREINCONSISTENT(Hyp,Th,CS,Δ-Set)
    TP ← Th                          ▷ Reinitialize the theorem prover only with the theory
    for all 𝒞𝒪 ∈ CS do
        if checkConsistency(𝒞𝒪, Hyp \ 𝒞𝒪) = ∅ then
            Δ-Set ← Δ-Set ∪ 𝒞𝒪          ▷ Add conflict to the diagnoses if Th ∪ 𝒞𝒪 ⊭ ⊥
        end if
    end for
    return Δ-Set
end function
```

Example 5.1 (cont.)

In this example, we show how abductive diagnoses can be obtained via HS-DAGXPLAIN for our running example. We can add the negated observation ¬*poor_lubrications* to the theory. We abbreviate the hypotheses for readability within the DAG, i.e., *Damaged_pump* $\equiv D$, *Broken_filter* $\equiv B$, *Cooler_leaks* $\equiv L$, *Cooler_cracks* $\equiv C$. Starting from the root in Figure 5.6, the first consistency check is performed. Since the edge label of the root node is empty, i.e., $h(root) = \emptyset$, all propositions in *Hyp* are assumed true, i.e., `checkConsistency`$(\{D, B, C, L\}, \emptyset)$. This of course leads to a contradiction, and *TP* returns the first conflict $\mathcal{CO}_1 = D$. $\mathcal{CO}_1$ is subsequently added to the set *CS* that records all refutations derived during computation.

Since the conflict has a cardinality of 1, only one new node n_1 is created, such that its edge label is the union of the root's edge label and the conflict element, i.e., $h(n_1) = \emptyset \cup \{D\}$. Again the theorem prover is called with `checkConsistency`$(Hyp \setminus h(n), h(n))$ resulting in a new conflict $\mathcal{CO}_2 = \{B, C, L\}$ (see Figure 5.7). For each element in $\mathcal{CO}_2$ a new node is created and the edge label computed accordingly. The consistency check for all newly created nodes n_2 to n_4, determines that these are consistent. These nodes are marked with ✓ and since there are no more nodes to process the computation is finished. Thus, $CS = \{\{D\}, \{B, C, L\}\}$. In the last steps of the algorithm, it is checked whether $Th \cup \mathcal{CO}$ does not lead to a contradiction. Therefore, *TP* is reinitialized with the original theory and each derived conflict is checked for consistency. Since both conflicts are consistent given the initial theory, both constitute diagnoses.

`checkConsistency`$(\{D, B, C, L\}, \emptyset) = \{D\} = \mathcal{CO}_1$

Figure 5.6: Initial DAG.

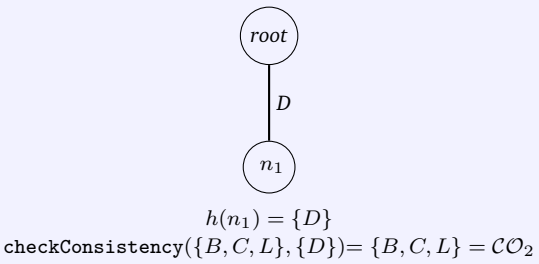


Figure 5.7: First level.

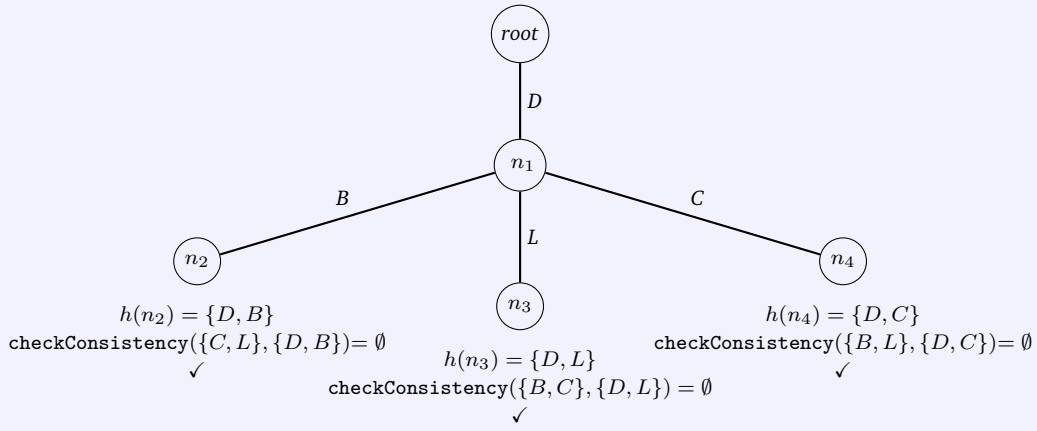


Figure 5.8: Second level.

As mentioned above, the conflicts returned by the theorem prover may not be parsimonious. Hence, given a prover that does not guarantee minimal conflicts, the computed explanations have to be reduced to retrieve parsimonious solutions. We propose two set-ups in this regard: (1) a simple method, which records the contradictions attained during the search and afterwards removes all non-minimal conflicts via subset checks. This basically corresponds to the conflict-driven HSDAGAB we have discussed on the previous chapter. (2) A new approach, which does not minimize the solutions at the end of the computation, but reduces each conflict returned by the theorem prover to a parsimonious one immediately. As a minimization procedure we can exploit, for example, Junker's [Jun04] QuickXplain. QuickXplain has been proposed as an efficient method to derive a single minimal preferred explanation for over-constrained systems based on a divide and conquer strategy. By recursively partitioning the set of constraints, the algorithm relaxes parts of the formulae while narrowing down on the set of constraints leading to an inconsistency. While there has been research suggesting the use of QuickXplain as the theorem prover for HS-DAG [FS10], our method is different as it derives the conflicts using a special purpose theorem prover and only exploits QuickXplain as a minimization procedure invoked on the already computed conflicting assumptions. Since we are considering QuickXplain in the context of diagnosis, we are not interested in conflicting formulae or constraints but abducibles. Hence, we propose an assumption-based version of the algorithm, which takes the *PHCAP* as well as one conflict $\mathcal{CO}$ returned by the theorem prover and extracts the minimal refutation based on $\mathcal{CO}$.

5.2.4 Conflict-Driven Search via Power Set Exploration

Within the context of infeasibility analysis conflicts are known as MUSes or unsatisfiable cores. Recall from the previous chapter that an MUS is a set of clauses that cannot be satisfied simultaneously, while every proper subset of an MUS is satisfiable [LS08]. Thus, in our proof-tree style abductive context, we want to compute contradictions, i.e., MUSes, from an rewritten inconsistent system representation. In the previous chapter, we have discussed the MUS enumeration tool MARCO [Lif+16] that extracts conflicts of an unsatisfiable formula based on the exploration of the power set of clauses. In this framework, the power lattice corresponds to an infeasibility map, which allows to avoid already investigated regions, and is encoded as a Boolean formula. The MUSes computation starts with a seed, i.e., a set of clauses from the map indicating an unexplored section. In case the seed is unsatisfiable, it is decreased to an MUS. To reduce the seed to a conflict any single-MUS extraction algorithm can be utilized. The simplest procedure descends the lattice until an MUS is identified, i.e., an unsatisfiable set of clauses, whose children are satisfiable. In case the seed is satisfiable, it is expanded until a maximal set of satisfiable clauses, an MSS, is determined. Again the simplest method identifies the maximum satisfiable superset by traversing the lattice upwards. Once an MUS or MSS has been found, clauses are added to the encoding of the map in order to exclude supersets (subsets) of already found MUSes (MSSes), since given an unsatisfiable (satisfiable) clause set, all of its supersets (subsets) are unsatisfiable (satisfiable) as well. Thus, an MUS defines a “low point" in an infeasible region, while an MSS is a “high point" in a satisfiable region. The process is restarted by determining a new seed given the map encoding. Once the entire map has been inspected this way, it is ensured that all MUSes and MSSes have been recovered.

This approach has been applied to axiom pinpointing for lightweight description logics, where it has been modeled as the enumeration of group-MUSes[4] of Horn formulas [Ari+15]. To boost performance, this version favors unsatisfiable seeds by computing maximal models of the map, i.e., the maximum number of literals is *true* without violating a clause. This way MUSes are derived early on and every satisfiable seed is guaranteed to already constitute an MSS. Therefore, there is no need for an operation traversing the lattice upwards to identify an MSS. Further as Arif et al. [Ari+15] are dealing with Horn clauses, they utilize LTUR for satisfiability testing and present an insertion-based MUS extraction procedure using an incremental LTUR which allows to add clauses one at a time. They have shown that in the context of Horn formulae based on an incremental implementation of the LTUR algorithm the insertion-based MUS extraction procedure is a competitive alternative to other approaches.

To evaluate the traversal of the power set in the context of abductive model-based diagnosis we have adapted the power set approach. Instead of handling constraint sets or group-MUS, we use an assumption-based set-up. Thus, the generated conflicts only comprise our abducible hypotheses as already discussed for HS-DAG. Algorithm XPLORER depicts our method based on the procedure proposed by Liffiton et al. [Lif+16]. First, to create a contradicting formula, the theorem prover *TP* is supplied with the theory and negated observations. In our assumption-based version, the map holds only Boolean variables representing elements of *Hyp*. To bias the procedure towards MUSes, we compute the maximal model (`getMaxModel`) of the map for the seed [Ari+15]. This allows us to disregard a growing method and right away use any satisfiable seed for blocking down within the map. After computing a seed, *TP* is called to check the consistency of the seed in consideration of the theory and negated observations. In case the theorem prover is consistent, an MSS has been found (due to using a maximal model as seed). To ensure that the satisfiable seed

[4] Clauses are partitioned into disjoint sets (groups) and MUSes are expressed as sets of groups.

and its subsets are no longer considered in the proceeding computations a blocking clause has to be added to the map. The method `blockDown` creates a clause $B_{\downarrow} = \bigvee_{s \not\subseteq seed} s$. In case *TP* returns a refutation, an MUS is extracted by applying a shrinking procedure to the seed. Subsequently, any superset of the MUS are prevented from the computation by applying a blocking clause to the map, i.e., $B_{\uparrow} = \bigvee_{m \in MUS} \neg m$ via `blockUp`. As long as the map is satisfiable, a seed is generated from the map, otherwise all MUSes and MSSes have been found and the set of abductive diagnoses, i.e., MUSes, is returned.

***Algorithm 5.4:* XPLORER (based on [Lif+16])**

Require: *A*: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: Horn theory, *Obs*: the set of observations
Ensure: Δ-Set: set of minimal diagnoses

```
TP ← Th ∧ (⋀_{o∈Obs} o → ⊥)                          ▷ Creates contradictions
MUSes ← ∅
map ← BoolFormula(nvars = |Hyp|)     ▷ Empty Boolean formula with |Hyp| variables
while true do
    if map is satisfiable then
        seed ← getMaxModel(map)          ▷ Bias towards unsatisfiable seeds [Ari+15]
        CO ← checkConsistency(seed, Hyp \ seed)          ▷ Check seed's consistency
        if CO ≠ ∅ then                                          ▷ Seed is unsatisfiable
            MUS ← shrink(seed)                                    ▷ MUS extraction
            MUSes ← MUS ∪ MUSes
            map ← map ∧ blockUp(MUS)          ▷ Blocks MUS and all its all supersets
        else                                                      ▷ Seed is satisfiable
            map ← map ∧ blockDown(seed)          ▷ Blocks seed and all its subsets
        end if
    else
        Δ-Set ← MUSes
        return Δ-Set
    end if
end while
```

Example 5.1 (cont.)

Consider the running example[a]. First, the assumption-based theorem prover is set up with the theory and negated observations in addition to the construction of the the Boolean formula representing the map. In Figure 5.9 the initial unexplored power lattice featuring the variables from *Hyp* is depicted. Since the empty map is satisfiable, a seed is generated. In our example, the first seed is $\{D, B, L, C\}$, which is underlined within Figure 5.10.

Using *TP* we check whether assuming all hypotheses in the seed being true retains consistency. This is not the case, so the seed is reduced to an MUS, i.e., $\{D\}$ (framed within Figure 5.11). Afterwards, the blocking clause $\neg D$ is added to the map and thus all supersets of D will no longer be considered. Thus, the red portion of the lattice in Figure 5.11 depicts the already explored portion of the infeasibility map.

The next seed is $\{B, L, C\}$, which again is unsatisfiable (see Figure 5.12). When applying a shrinking function, $\{B, L, C\}$ proves to already constitute a minimal conflict and hence, is added to MUSes. Subsequently, `blockUp`$(\{B, L, C\})$ is appended to the

map blocking $\{B, L, C\}$ and $\{D, B, L, C\}$, which has already been marked as explored previously.

As shown in Figure 5.13 in the next step a new maximal model is retrieved, i.e., $\{B, C\}$. Since this seed is consistent, it represents an MSS. Hence, a clause blocking downwards $B_{\downarrow} = (D \vee L)$, which marks all subsets of the seed as explored, is added to the map. This process is repeated until the map is no longer satisfiable (shown in Figure 5.14). The resulting refutations and thus minimal diagnoses are MUSes $= \{\{D\}\{B, L, C\}\}$.

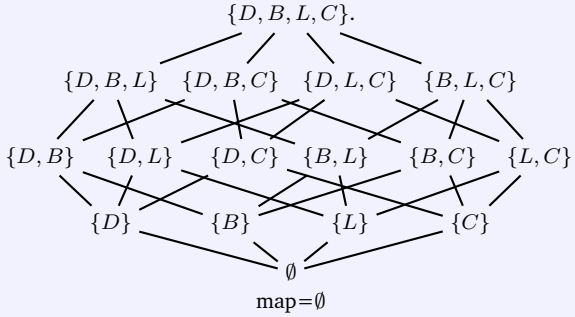


Figure 5.9: Initial map.

[a] Again for readability we abbreviate the hypotheses, i.e., *Damaged_pump* $\equiv D$, *Broken_filter* $\equiv B$, *Cooler_leaks* $\equiv L$, *Cooler_cracks* $\equiv C$.

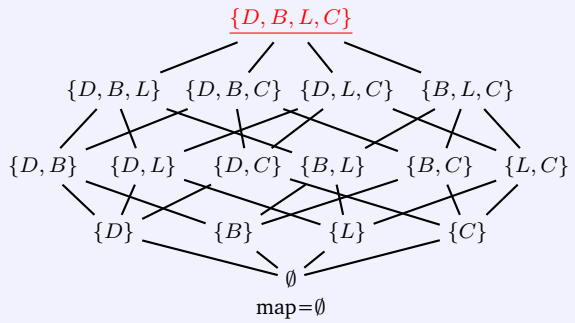


Figure 5.10: First maximum model seed $\{D, B, L, C\}$.

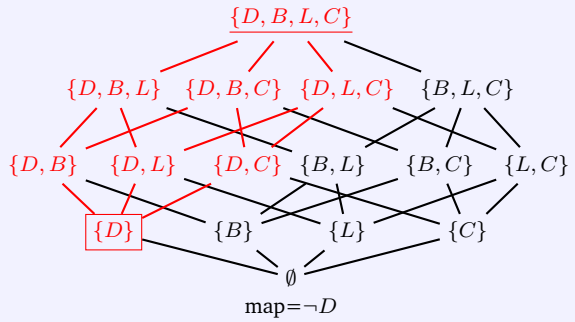
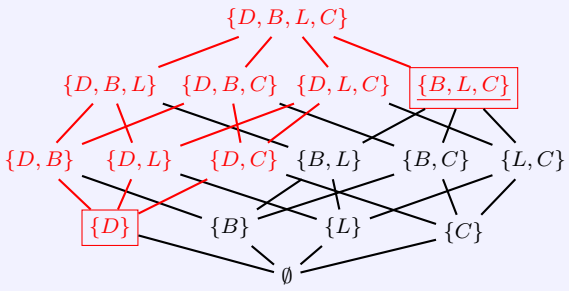
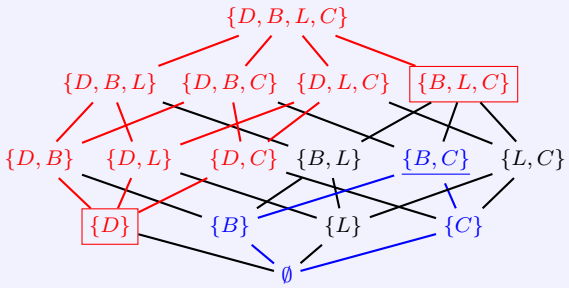


Figure 5.11: Shrinking $\{D, B, L, C\}$ to first the MUS $\{D\}$ and blocking all supersets of $\{D\}$.



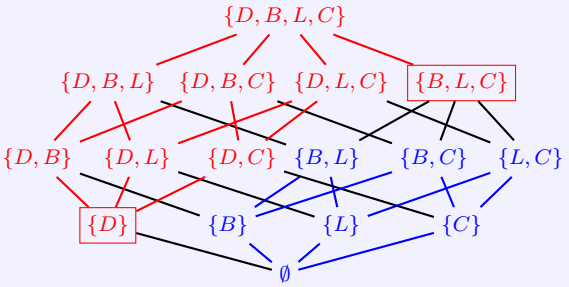
map=$\neg D \wedge (\neg B \vee \neg L \vee \neg C)$

Figure 5.12: Second seed/MUS $\{B, L, C\}$ and blocking all supersets of $\{B, L, C\}$.



map=$\neg D \wedge (\neg B \vee \neg L \vee \neg C) \wedge (D \vee L)$

Figure 5.13: Third seed $\{B, C\}$ and blocking all subsets of $\{B, C\}$.



map=$\neg D \wedge (\neg B \vee \neg L \vee \neg C) \wedge (D \vee L) \wedge (D \vee C) \wedge (D \vee B)$

Figure 5.14: Final map.

5.2.5 Abduction under Stable Model Semantics

In ASP solutions to hard search problems are derived based on stable model semantics of logic programming. Problems are represented declaratively as logic programs, while their solutions are equivalent to the logic program's answer sets. A normal logic program Π consists of a finite set of rules of the form $p_0 \leftarrow p_1, \ldots, p_m, \neg p_{m+1}, \ldots, \neg p_n$, where each p_i is an atom. Given such a logic program its answer sets can be obtained by creating the Gelfond-Lifschitz reduct of the original program and computing the models on the reduct [Ang+05] . The reduct $\Pi_{\mathcal{M}}$ of a normal logic program Π relative to a set of atoms $\mathcal{M}$ can be derived by removing

1. each rule with $\neg l$ in its body, where $l \in \mathcal{M}$ and
2. all negative literals in the bodies of the remaining rules [Eit+09].

If $Cn(\Pi_{\mathcal{M}})$, the $\subseteq$-smallest model of the reduct $\Pi_{\mathcal{M}}$, corresponds to $\mathcal{M}$, then $\mathcal{M}$ is a stable model, i.e., answer set, of Π [Ang+05].

Example 5.2 **[Ang+05]**

Consider the logic program

$$\Pi = \left\{ \begin{array}{ll} (1) & x \leftarrow x, \\ (2) & y \leftarrow \neg x \end{array} \right\}.$$

To derive the answer sets we start with $\mathcal{M} = \emptyset$ and create the reduct $\Pi_\emptyset$ (see Table 5.1). Given $\mathcal{M} = \emptyset$, the first condition does not apply, but we can remove the negative literal $\neg x$ from the second rule, resulting in $\Pi_\emptyset = \{(1)\ x \leftarrow x,\ (2)\ y \leftarrow\}$. Based on $\Pi_\emptyset$ we can derive $Cn(\Pi_\emptyset) = \{y\}$. Yet, since $\{y\} \neq \emptyset$, i.e., $Cn(\Pi_\mathcal{M}) \neq \mathcal{M}$, $\emptyset$ is not an answer set.

Consider the second row in Table 5.1, where $\mathcal{M} = \{x\}$. The first condition applies and thus the second rule $y \leftarrow \neg x$ is removed from the program resulting in the reduct $\Pi_{\{x\}}$. However, as $Cn(\Pi_{\{x\}}) \neq \{x\}$, x does not correspond to an answer set. For $\mathcal{M} = \{y\}$ the situation is different; the reduct's model $Cn(\Pi_{\{y\}})$ and the set of atoms in $\mathcal{M}$ match. Hence, $\{y\}$ is an answer set for Π. More specifically, it constitutes the only answer set for the logic program.

Table 5.1: Computation of the answer sets.

$\mathcal{M}$	$\Pi_\mathcal{M}$	$Cn(\Pi_\mathcal{M})$	$Cn(\Pi_\mathcal{M}) = \mathcal{M}$
$\emptyset$	$\{(1)\ x \leftarrow x\ ,\ (2)\ y \leftarrow\}$	$\{y\}$	✗
$\{x\}$	$\{(1)\ x \leftarrow x\}$	$\emptyset$	✗
$\{y\}$	$\{(1)\ x \leftarrow x\ ,\ (2)\ y \leftarrow\}$	$\{y\}$	✓
$\{x, y\}$	$\{(1)\ x \leftarrow x\}$	$\emptyset$	✗

ASP has been applied to diverse domains such as planning [Lif02]. Furthermore, ASP under stable model semantics can be used for abductive reasoning. By generating an encoding for propositional abduction using a *guess-and-test* strategy, rules and constraints determine the search space for answers (*guess*) and define whether generated solutions are valid (*test*). Saikko, Wallner, and Järvisalo [Sai+16] have proposed such an encoding of propositional abduction which generates minimum-cost solutions via the negation of the conjunction of observations. Their logic program guesses a candidate solution from the set of hypotheses and checks if it entails the manifestations and is consistent given the background theory based on a saturation technique.

5.3 Empirical Results

In this section, we present our empirical evaluation framework and report on the obtained results. Our focus lies on identifying runtime trends suggesting the advantage of one approach over the other. Further, we want to contribute to answering the questions (1) whether a general-purpose solver can be used for an efficient diagnosis in practice or if a specifically tailored engine is required for achieving convincing performance and (2) whether there is a superiority of direct or conflict driven methods.

First, we give some insights into the implementations of the abduction methods discussed in the previous section. Then, we report on the benchmarks considered in the experiments.

Particularly, we utilized artificially generated diagnosis problems as well as real world failure models. Subsequently, we discuss the results of the experiments all obtained from a Mac Pro (Late 2013) with a 2.7 GHz 12-Core Intel Xeon ES processor and 64GB of RAM running OS X 10.10.5.

5.3.1 Algorithms

To compare the abductive diagnosis approaches explained in Section 5.2, we implemented parts of the methods in Java as well as exploited available tools realizing the abductive reasoning procedures. The following implementations are used in the empirical evaluation:

- **Abduction with the ATMS** (*ATMS*): Within our evaluation we exploit a Java implementation of ATMSXPLAIN based on a Java implementation of an ATMS in its original form, i.e., restricted to Horn theories. Since we are dealing with Horn clauses, we utilize LTUR as the theorem prover allowing us to compute inferences efficiently on our models. In particular, we exploit a Java implementation of an assumption-based LTUR of the diagnosis engine jdiagengine[5] [PW03]. We refer to this entire abduction procedure simply as *ATMS* within our experiments.

- **Abduction as Consequence Finding via SOL-resolution** (*CF*): We created a consequence finding set-up *CF* using a Java implementation of CONSQXPLAIN. To enumerate the characteristic clauses we employ SOLAR[6] [Nab+10], which is a Java implementation of SOL-tableaux calculus for first order full clausal theories. The tool provides various pruning methods, which ensure, for instance, that redundant tableaux are not generated. We invoked SOLAR with the default setting, which executes a depth-first search within the tableaux repeatedly, incrementing the depth limit for each iteration.

- **Conflict-Driven Search via HS-DAG**: We use jdiagengine [PW03], which implements a conflict-driven search via HS-DAG coupled with an incremental assumption-based LTUR, to realize HS-DAGXPLAIN. LTUR does not guarantee to return minimal conflicts, hence the contradicting assumptions have to be reduced to derive minimal explanations. We use two set-ups:

  - *HS-DAG*: For this procedure, we simply record all refutations returned within the search on the HS-DAG. After the computation has finished, all conflict supersets are removed resulting in parsimonious diagnoses.

  - *HS-DAG$_{QX}$*: We adapted jdiagengine's implementation to minimize refutations right away after being returned from LTUR. In particular, we created a Java implementation of QuickXplain to reduce each conflict to a parsimonious one. Our version of QuickXplain is based on assumptions rather than on constraints with preferences and exploits the assumption-based LTUR for its consistency checks. Every refutation minimized by QuickXplain already constitutes an abductive explanation.

- **Conflict-Driven Search via Power Set Exploration**: We realized XPLORER in Java. To favor unsatisfiable cores early on in the computation, we implemented Arif et al. 's [Ari+15] method to find maximal model seeds. In addition, we used the SAT solver SAT4J[7] [ES03] in order to determine the satisfiability of the Boolean formula representing the map. To shrink a found unsatisfiable seed to an MUS, we use two extraction mechanisms:

[5] `www.ist.tugraz.at/modremas/index.html`
[6] Used version: SOLAR 2 `(Build 315)`
[7] `www.sat4j.org/`

– *XPLorer*: For our first set-up we implemented the insertion-based MUS extraction algorithm as suggested by Arif et al. [Ari+15] using jdiagengine's incremental assumption-based LTUR.

– *XPLorer*$_{QX}$: Since each unsatisfiable seed already constitutes a conflict and the MUS extraction merely reduces it towards a minimal contradiction, we again applied the assumption-based QuickXplain implementation to minimize the unsatisfiable seed to an MUS, i.e., abductive explanation.

- **Abduction under Stable Model Semantics** (*ASP*): By exploiting an encoding of propositional abduction [Sai+16][8] we compute diagnoses via the state-of-the-art C++ ASP solver clingo 4.5.4[9] [Geb+14]. The encoding generates minimum-cost solutions using the negation of the conjunction of observations based on a technique called saturation to determine whether the entailment relation between diagnoses and manifestations exists. As we want to enumerate all subset minimal answer sets, we remove the optimization criteria included in the encoding and apply domain-specific heuristics to the solver over the command line[10]. We call this approach *ASP* within the evaluation.

In *CF* and *ASP*, we invoke both general-purpose solvers using a separate process inside our implementation, even though we could use SOLAR directly in our Java code. We use this set-up to mimic that both tools are independent from the used programming language of the abduction procedure and simply function as black boxes returning the diagnoses.

5.3.2 Data

To asses the various abductive reasoning mechanisms presented, we chose to use two different types of models. On the one hand, we created artificial Horn clause models, which are similar to diagnosis knowledge used in practice. On the other hand, we exploit knowledge on how component-based failures affect real world systems, i.e., FMEAs. The information is automatically compiled into a Horn theory via a mapping function and subsequently used in the context of abductive model-based diagnosis.

Artificial Benchmarks

By means of a sample generator, we constructed artificial *PHCAPs*. The rules contained within the theory are based on several parameters; n is the minimum number of literals in a rule's body. Those literals are chosen randomly from A. Thus, this value indirectly determines the number of hypotheses present within the entire model. r bounds the number of clauses generated per body, while o limits the overlap between head literals. The head of each rule can only contain elements from $A \setminus Hyp$, thus, we do not have assumptions that are caused by another hypothesis. Further, the fabricated models do not contain any cycles. The parameter k determines the maximum number of manifestations to be explained. Those observations are randomly selected from $A \setminus Hyp$ and for simplicity comprise only positive propositions. Figure 5.15 depicts the principle structure of these artificial examples as an And-or-graph, where we denote hypothesis nodes with H_i and nodes representing propositions from $A \setminus Hyp$ with e_j.

The examples constructed by the generator are similar to real world samples. Just consider, for instance, medical diagnosis, where the knowledge mainly captures how a set of

[8] `www.cs.helsinki.fi/group/coreo/abhs/`
[9] `www.potassco.org/clingo/`
[10] clingo is invoked with `- -heuristic=Domain`, `- -enum-mod=domRec`, and `- -dom-mod=5,16`. This ensures the computation of subset minimal answer sets.

assumptions about diseases affects a set of symptoms. Usually, we cannot observe cycles or an excessive number of implication levels within this type of knowledge.

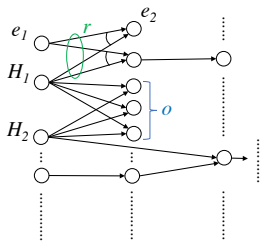


Figure 5.15: Structure of the artificial examples used for evaluation.

For the empirical evaluation we fabricated two different sets of artificial samples; *Artificial Samples I* was constructed with $k{=}5$, $r{=}o{=}15$ and $n{=}1$ and contains 166 *PHCAPs*. For *Artificial Samples II* the example generator was invoked with $k{=}o{=}r{=}5$ and $n{=}1$ and created 118 diagnosis problems. Table 5.2 summarizes the statistics for the generated Horn models.

Table 5.2: Sample set statistics for the artificial benchmarks.

	Artificial Samples I				*Artificial Samples II*			
	MIN	**MAX**	**AVG**	**MED**	**MIN**	**MAX**	**AVG**	**MED**
$\|Hyp\|$	10	504	275.07	320.00	12	235	120.42	112.50
$\|A \backslash Hyp\|$	6	6,466	1,903.23	1,668.00	13	1,055	252.74	180.00
$\|Th\|$	10	7,186	2,950.10	2,731.00	20	1,146	416.70	358.50
$\|Obs\|$	1	5	2.86	3.00	1	5	2.72	2.50
$\|\Delta$-Set$\|$	1	50	2.76	1.00	1	58,520	500.71	2.00

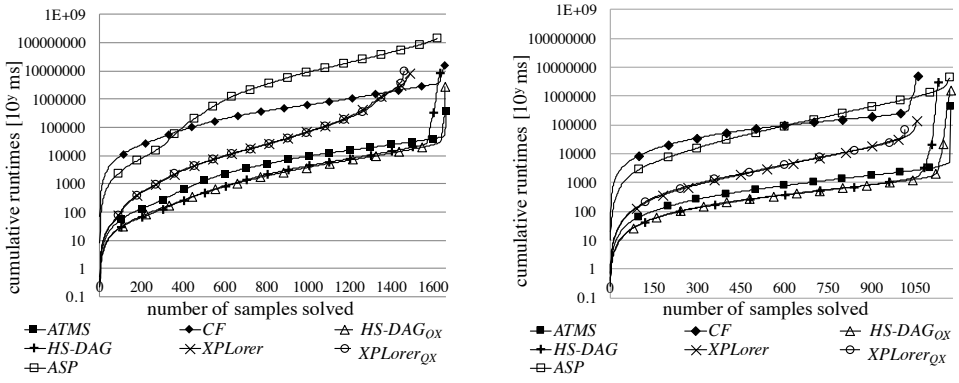
Real World Samples

To make use of abductive inference in industrial applications, Wotawa [Wot14] proposes to automatically extract the required system descriptions from failure assessments which are commonly used in practice. In particular, FMEA provides all information necessary to function as a basis of an abductive diagnosis model.

For our experiments, we again utilize several publicly available as well as project internal FMEAs comprising diverse technical systems and sub-systems. Based on the models we created 213 diagnosis problems, where we randomly chose observations from the set of effects described in the assessment. The statistical information on the FMEA models is shown in Table 5.3.

Table 5.3: Sample set statistics for the FMEA benchmark.

	FMEA Samples			
	MIN	**MAX**	**AVG**	**MED**
$\|Hyp\|$	3	90	26.16	20.00
$\|A \backslash Hyp\|$	5	83	26.60	21.00
$\|Th\|$	12	298	70.59	37.00
$\|Obs\|$	1	29	10.79	10.00
$\|\Delta$-Set$\|$	1	2,288	67.98	6.00



(a) *Artificial Samples I* **(b)** *Artificial Samples II*

Figure 5.16: Numbers of diagnosis samples solved over time for the artificially generated diagnosis problems.

5.3.3 Results

As mentioned, all algorithms and tools are implemented in Java, except clingo, which is a C++ ASP solver. Each method was invoked ten times on each *PHCAP*; for instance, we collected 1,660 data points per abductive reasoning approach in the context of *Artificial Samples I*. Since we are interested in runtime trends and trade-offs between the algorithms, our evaluation focuses on computation times. Note here that we record the time to extract all minimal explanations. Although we could use some of the approaches to derive a partial set of solutions, we are not considering this possibility in the evaluation. In addition, we disregard the rewriting of the model, the creation of the different input formats, and the time it requires to communicate with the solvers. In case of SOLAR and clingo, we parse the execution times recorded by the tools themselves which are available in the output[11].

Table 5.4 gives an overview of the evaluation results on all sample sets for our seven abduction methods. For clarity, we put the best values per category in bold face. Each computation faced a runtime limit of twenty minutes. In the rows categorized *Runtimes* we provide the performance statistics based on the samples that were computed in time, i.e., all executions where the approach exceeded the limit are not considered within these numbers. The number of samples completed in time is reported in Table 5.4 in row **# solved**. For example, for *Artificial Samples I* only *ATMS* managed to derive all solutions for all executions within the given time frame. Additionally, we report the number of samples an approach has computed the diagnoses the fastest in row **# fastest**. In contrast to *Runtimes*, $Runtimes_\theta$ contains results where each timed out execution is penalized with $\theta = 40$ *minutes*[12]. We do not report on the minimal result values in $Runtimes_\theta$, as they are equivalent to *Runtimes*, nor on the maximum execution times which are either the same as for *Runtimes* or the penalized runtime.

Artificial Benchmarks

To illustrate the runtimes of the algorithms, we ordered all successful sample runs according to their execution time. In Figure 5.16a and Figure 5.16b we report on the number of samples solved for growing cumulative log runtime for *Artificial Samples I* and *Artificial Samples II*, respectively. From the graphics in Figure 5.16 we deduce that all algorithms

[11] We converted the values to milliseconds for our analysis.

[12] We set θ to twice the runtime threshold. Of course, this value is somewhat arbitrary, since we do not know the real computation time of an approach exceeding the time limit.

Table 5.4: Runtime results.

			ATMS	*CF*	*HS-DAG*	*HS-DAG$_{QX}$*	*XPLorer*	*XPLorer$_{QX}$*	*ASP*
Artificial Samples I [1,660 runs]	**Runtimes [in ms]**	**# solved**	**1,660**	1,650	1,629	1,654	1,485	1,457	1,618
		# fastest	183	0	**820**	629	15	13	0
		MIN	0.34	70.00	0.23	**0.22**	0.49	0.35	8.00
		MAX	**170,838.66**	899,370.00	1,177,876.77	952,760.53	567,187.64	904,397.79	1,156,071.00
		AVG	**226.28**	9,387.55	5,397.07	1,600.36	5,466.45	6,590.33	88,315.45
		MED	19.91	1,080.00	9.20	**8.04**	88.96	80.22	17,931.50
		SD	**4,529.92**	65,366.46	52,305.30	34,443.66	27,536.54	48,291.29	174,211.35
	Runtimes$_\theta$ [in ms]	**AVG**	**226.28**	23,788.83	50,115.55	10,269.27	257,902.21	299,278.38	146,803.85
		MED	19.91	1,110.00	9.76	**8.13**	139.55	130.70	20,678.50
		SD	**4,529.92**	196,183.91	328,376.83	148,023.42	736,034.18	785,667.50	401,804.10
Artificial Samples II [1,180 runs]	**Runtimes [in ms]**	**# solved**	**1,171**	1,139	1,130	1,170	1,055	1,015	1,168
		# fastest	231	0	**534**	415	0	0	0
		MIN	0.39	80.00	0.26	**0.25**	0.69	0.72	10.00
		MAX	425,393.84	389,900.00	391,309.67	217,275.60	27,919.00	**27,649.77**	315,250.00
		AVG	367.55	4,941.55	2,783.68	1,376.12	132.65	**67.36**	4,031.42
		MED	2.23	220.00	**0.99**	1.00	14.13	12.88	470.00
		SD	12,431.07	33,547.10	27,328.01	12,864.20	1,048.81	**870.91**	22,215.67
	Runtimes$_\theta$ [in ms]	**AVG**	**9,517.29**	46,464.77	53,513.19	11,533.95	127,237.24	167,854.55	16,193.81
		MED	2.26	240.00	1.02	**1.00**	20.45	19.49	490.00
		SD	**105,143.70**	221,759.43	242,745.11	110,664.17	369,418.07	416,327.19	122,059.98
FMEA Samples [2,130 runs]	**Runtimes [in ms]**	**# solved**	2,119	**2,130**	1,758	2,020	1,918	1,650	2,020
		# fastest	**1,267**	19	294	539	8	3	0
		MIN	0.33	60.00	**0.24**	0.25	0.56	0.56	6.00
		MAX	916,524.17	375,020.00	1,191,864.18	140,108.57	883,309.51	903,439.88	**14,680.00**
		AVG	1,854.24	2,375.50	22,709.35	641.59	10,685.16	5,662.87	**239.74**
		MED	**0.90**	200.00	1.87	1.26	23.36	18.05	32.00
		SD	30,796.96	24,030.25	111,354.35	6,775.37	68,640.88	47,594.18	**1,299.25**
	Runtimes$_\theta$ [in ms]	**AVG**	14,239.04	**2,375.50**	437,898.14	124,552.12	248,494.90	545,231.80	124,171.03
		MED	**0.91**	200.00	4.25	1.39	29.77	30.49	35.00
		SD	174,655.60	**24,030.25**	908,437.40	531,157.15	718,422.99	1,001,498.81	531,206.63

experience an exponential runtime curve. In addition, the plots show that the same technique with different minimization algorithms, i.e., *XPLorer* and $XPLorer_{QX}$ as well as *HS-DAG* and $HS\text{-}DAG_{QX}$, feature a similar performance and only diverge in regard to their efficiency towards the more runtime expensive instances.

ATMS is capable of solving the most *PHCAPs* within the given time limit for both sample sets, followed by $HS\text{-}DAG_{QX}$. In contrast, approaches based on the exploration of the power set fail to compute solutions for up to thirteen percent of the examples in time. Hence, the execution times in *Runtimes* in Table 5.4 can be deceiving since they, for instance, might convey that $XPLorer_{QX}$ provides preferable results in regard to the maximum and average runtime. Yet considering the penalized results, we can see that the approach is not competitive. From the penalized runtimes, we deduce that *ATMS* and $HS\text{-}DAG_{QX}$ are superior in comparison to the remaining techniques.

Unsurprisingly, the non-Horn reasoning tools, i.e., clingo and SOLAR, are about two orders of magnitude slower than the fastest approach on the artificial examples. Although the methods are inefficient on average, the results show that the applications are dependable methods by solving between 97 % to 99 % of the samples in time for both evaluation sets. From Figure 5.16, we can further conclude that for the simpler examples the computation time for *CF* grows faster than for the ASP solver. This situation, however, changes with more runtime expensive diagnosis problems.

For a more in-depth comparison of the algorithms, we created the scatter plots in Figures 5.17 and 5.18. Each data point symbolizes one sample run based on the penalized log runtimes of the corresponding algorithm pair. The dashed lines mark the penalized runtime, i.e., every execution exceeding the runtime limit is located on the dashed lines. Consider Figure 5.18a; although there are executions where $HS\text{-}DAG_{QX}$ is rather inefficient on *Artificial Samples II* in comparison to *ATMS*, the data points accumulate on and below the diagonal close to the origin. This suggests that for the bulk of the samples, *ATMS* takes longer to compute the diagnose; this is also indicated by the median runtime results in Table 5.4 under $Runtimes_{\theta}$. Investigating the threshold lines, we can determine that only a single execution led to a timeout on $HS\text{-}DAG_{QX}$ but not on *ATMS*. All other timed-out data points are exactly at the intersection of the threshold lines, thus representing several instances causing both approaches to exceed the time limit.

While $HS\text{-}DAG_{QX}$ provides more appealing results in comparison to the version without QuickXplain, we cannot observe the same for *XPLorer* and $XPLorer_{QX}$. There both conflict extraction methods perform rather similarly. This is apparent from Figures 5.17d and 5.18d, where data points are mostly located symmetrically around the diagonal. Yet the insertion-based extraction of the conflict as suggested by Arif et al.'s [Ari+15] solves more samples in time. Specifically, on *Artificial Samples II* only a few samples exist in which *XPLorer* was penalized while $XPLorer_{QX}$ provided the solutions in time.

For the two non-Horn solvers, the results on the artificial samples are interesting. On the first set of examples, the consequence finding procedure is preferable, i.e., more samples are solved in time and on median consequence finding is more efficient than using the ASP solver. This is also apparent from Figure 5.17f. On *Artificial Samples II*, yet, the situation changes slightly since ASP can solve more samples in time than the *CF* method and provides better average results. Examining the minimum runtimes, we can deduce that the two approaches using off-the-shelfs solvers suffer from a computational overhead. Particularly, there seems to be even more set-up effort required for SOLAR[13] in comparison to clingo.

[13]We extract the execution times directly from the solvers' output themselves.

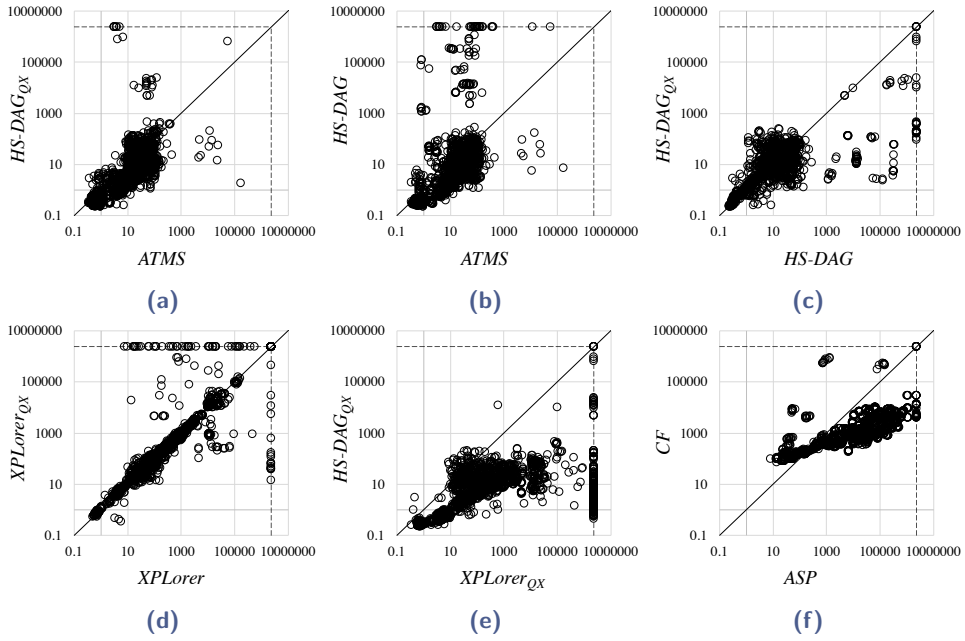


Figure 5.17: Scatter plots comparing the penalized runtimes [10^y ms] on *Artificial Samples I*.

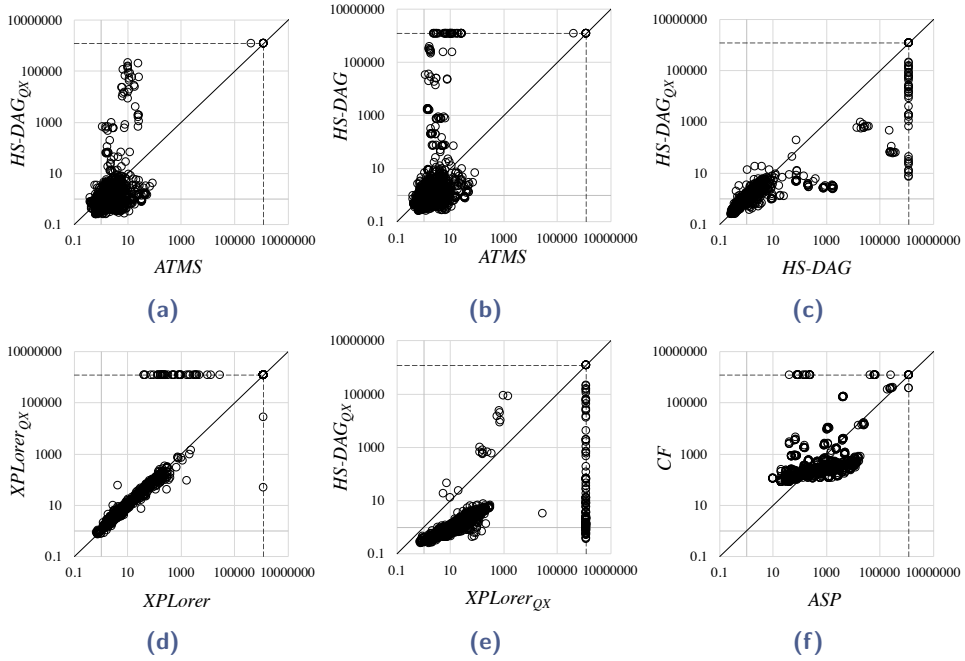


Figure 5.18: Scatter plots comparing the penalized runtimes [10^y ms] on *Artificial Samples II*.

Real World Samples

Reviewing the computation results for the FMEA-based examples the consequence finding approach is the only method solving all 2,130 executions within the given time frame. This subsequently causes it to provide good results within $Runtimes_\theta$ since no additional penalty is added. Figure 5.19, which displays the cumulative runtimes without penalty, shows that the method is not competitive in comparison to the faster procedures such as abduction with the *ATMS* or the HS-DAG approaches. On the FMEA samples, the superiority of *ATMS* in comparison to $HS\text{-}DAG_{QX}$ and the remaining approaches is more noticeable than on the artificial benchmarks.

In addition, we can observe in Figure 5.20c and 5.20d that the computation times of the variations of HS-DAG and power lattice exploration diverge more in comparison to the artificial samples. Given the runtime plot in Figure 5.19, we can determine that the divergence mainly occurs in the last third of samples solved.

Regarding this portion of the graph, we can further discover that *ASP*'s and *CF*'s execution times do not grow as steeply at the end as for the other approaches. Further, in Figure 5.20f the bulk of data points is located above the diagonal, indicating more samples in which *ASP* was superior. However, several examples could not be solved by *ASP* in time, hence, the less convincing results in $Runtimes_\theta$.

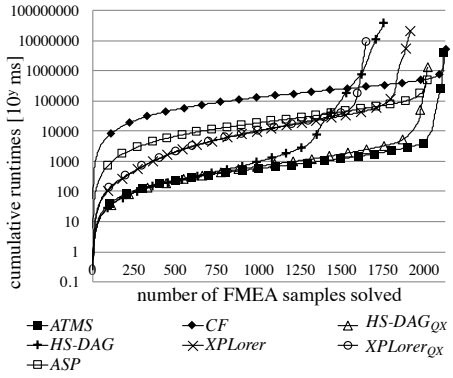


Figure 5.19: Numbers of diagnosis samples solved over time for the FMEA diagnosis problems.

5.3.4 Discussion

Considering the experiments, two approaches seem superior: *ATMS* and $HS\text{-}DAG_{QX}$. While *ATMS* shows promising results in our set-up, it has to warrant with each added clause all necessary vertex labels are updated, thus, labels have to be checked for consistency[9] and minimality. This is a time-consuming operation especially given a greater number of hypotheses and overlap, which both may lead to larger labels. Hence, for benchmarks with a vast number of assumptions, we assume less convincing results from *ATMS*. Comparing the FMEA-based to the artificial samples, the ATMS has to perform more propagations within the latter due the fact that the there might be several levels within the graph, while for the FMEA examples due to their structure it is ensured that there are at most three levels with the last level only containing the explanation node *ex*. In addition, the environments themselves consist of a single element (with the exception of the node *ex*), due to the bijunctive Horn clauses comprising the theory. Yet the labels themselves can become quite large depending on the interconnectedness of the And-or-graph, i.e., manifestations with a large number of

[9]Though in our case no conflicting hypotheses were generated within the PHCAPs.

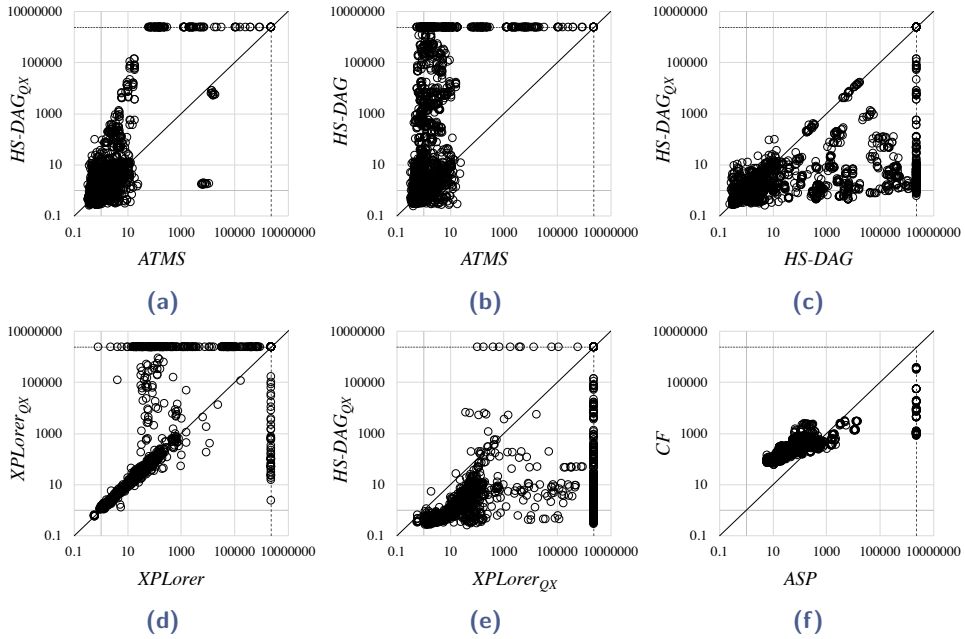


Figure 5.20: Scatter plots comparing the penalized runtimes [10^y ms] on *FMEA Samples*.

causes feature a large label, which has to be considered during minimization and consistency checks. Approaches to focus the ATMS and subsequently avoid label explosion have been proposed in the literature [FK88].

As a side note we would like to mention that the ATMS can also be exploited within a conflict-directed set-up. In fact in the consistency-based variation of model-based diagnosis, the ATMS records the inconsistencies arising from the health assumptions and observations. Thus, we could use a proof-tree completion approach to derive contradictions recorded within the `NOGOOD` node of the ATMS. However, to retain consistency, the ATMS has to warrant that each node's label does not contain the conflicts stored in `NOGOOD` or their supersets. Thus, more consistency checks are necessary in a proof-tree-style abduction with the ATMS than in our current experimental set-up where the `NOGOOD` node has a small or given our benchmarks even an empty label.

While showing weak runtime results on the artificial samples, SOL-resolution with SOLAR provides solutions for all *PHCAPs* within *FMEA Samples*. Given the simple structure of the FMEA examples the number of inference steps in the SOL-resolution are diminished in comparison to the artificial models. In particular, for one tableau the number of *resolve*, *skip*, *factor* and *truncation* steps is proportional to the number of observations, while for the artificial samples more reasoning steps may be necessary. Of course the number of tableaux depends on the number of hypotheses inferring the observations. As already mentioned, SOLAR requires some additional set-up time, which is unsurprising since it is not a specialized Horn abduction solver, but designed for first-order clausal theories. This observation relates to clingo, which is a powerful tool suitable for normal as well as disjunctive logic programs. Therefore, within our evaluation set-up abduction with the ASP solver is not competitive. From our analysis we know that the preparation (grounding) and preprocessing time (program simplification) are not the driving factors in the computational effort, but the true solving time is problematic from an efficiency point of view. We assume that we can observe particularly bad results in *Artificial Samples I* due the extensive number of variables, since the saturation technique is based on all variables within the *PHCAP*.

HS-DAGXPLAIN as well as XPLORER, both restrict the search space by ensuring that known conflicts and their supersets are not considered again during computation. In case of the former, the construction of a pruned DAG already hinders redundant refutations, while blocking clauses are added to the encoding of the infeasibility map in the latter. Another optimization of XPLORER is to require the seed of the lattice traversal to be a maximal model. This strategy favors contradictions and ensures that the entry point of the conflict search is as “high" as possible in the power set. In contrast in the HS-DAG approach, LTUR generates refutations, which are located somewhere between this maximum sized conflict, as enforced by XPLORER, and a minimal one. Hence, the conflict-driven search via HS-DAG requires less minimization effort than the power set exploration since the starting point of the traversal is “lower". Besides the shrinking of a contradicting set of assumptions to an MUS, which is affected by the size of the diagnosis set, XPLORER spends considerable time to determine the maximal model for the seed. In general, all these factor contribute to the inefficiency of XPLORER within our experiments. A remedy for this might be to use an indirect MUS approach, which first computes the set of MCSes and afterwards derives their minimal hitting sets. These methods have been shown to be efficient for the complete enumeration of refutations [LS08]. Taking a look back at our evaluation in the previous chapter, we could observe the same situation. There we contrast the MUS-enumeration tool MARCO and a method that first computes MCSes using MCS_{LS} and subsequently derives the conflicts as the hitting sets on the MCSes. In comparison, the MUS extraction could not compete with the indirect approach.

From the results, we conclude that our HS-DAGXPLAIN approach is performing well on the samples, in particular the version which minimizes conflicts right away. Generally, HS-DAGXPLAIN’s performance is affected by the size of Δ-Set , i.e., the number of refutations, and the size of the conflicts. Reason being that these two features determine the depth and breadth of the DAG; that is, the larger the conflicts, the more outgoing edges the conflict node has and hence the more child nodes have to be checked for consistency to determine whether they are minimal hitting sets or not. Considering Figure 5.18c, *HS-DAG*$_{QX}$ yields more convincing runtime results than *HS-DAG*. By shrinking each contradiction before continuing with the execution on the HS-DAG, the depth of the graph can be reduced as already computed conflicts and their supersets are excluded from further considerations due to the search pattern encoded in the DAG. Thus, the DAG constructed via *HS-DAG* can be larger with more conflicts than the one generated with *HS-DAG*$_{QX}$. Further, *HS-DAG*$_{QX}$ is advantageous in the sense that it can provide results even though the execution has not been concluded, since all conflicts returned already constitute minimal abductive diagnoses given they are consistent. However, as QuickXplain still requires calls to a theorem prover, in our case the assumption-based LTUR, the overall number of consistency checks increases in comparison to *HS-DAG*. Particularly in the case that the theorem prover returns an already minimal conflict invoking QuickXplain causes unnecessary consistency checks. Utilizing an approach which already derives minimal conflicts makes the minimization step obsolete. For instance, instead of using LTUR as the theorem prover to infer refutations, we could exploit QuickXplain right away, which returns one minimal conflict [Fel+13][14]. Adaptation and improvements of QuickXplain have been proposed such as MergeXplain [Shc+15], which returns a set of minimal conflicts. Other possible improvements encompass to adapt HS-DAGXPLAIN to a parallelized version, which significantly improvements performance [Jan+15].

Given the structural characteristics of the FMEA-based models we can easily represent the theory as a DAG with a forward structure from causes to effects. As we know this structure is in fact equivalent to the problems in the simple parsimonious set covering theory proposed

[14] In this case, we could still use LTUR or a SAT solver as QuickXplain’s consistency check mechanism.

by Peng and Reggia [PR90] and hence these types of diagnosis problems can be solved quite efficiently by employing hitting set algorithms as shown in Chapter 4.

5.4 Conclusion

Abductive reasoning is an intractable problem in general. Hence, determining algorithms that can compute explanations in an acceptable time frame is an interesting research problem with relevance not only for the diagnosis community, but also all other application areas of abduction. We tackle this problem in this paper by reviewing two problem formulations within the context of propositional Horn clause abduction, i.e., direct proof and conflict-driven methods. We have exploited well-known as well as recent abductive reasoning algorithms and have shown possible adaptations of the techniques to enable a more efficient computation given our context. Besides implementing abductive reasoning procedures, we have also included preexisting tools to give a sense of their capability in regard to this specific framework. To reveal performance trends, we created an evaluation set-up based on artificially samples similar to fault knowledge in practice and real world examples stemming from failure assessments.

Our results show that neither the direct nor the conflict-driven approach provides a universal advantage for Horn clause abduction. The two most promising techniques based on our created problem instances were *ATMS* and abduction via HS-DAG with an immediate minimization of conflicts via QuickXplain. These two methods are particularly suitable to compare direct and conflict-driven techniques, since they exploit the same theorem prover. Both could solve a reasonable amount of samples within the given time frame and our data shows that *ATMS* on average is the most efficient approach, while *HS-DAG*$_{QX}$ computes diagnoses faster for more instances. Even though the general solvers were around two orders of magnitude slower than the best Horn reasoner, they have shown a consistent performance being able to compute explanations for most samples within the given runtime allowance. These results demonstrate that off-the-shelf tools can very well function as abduction engines despite not being competitive in regard to runtime.

Inspired by infeasibility analysis, we implemented an exploration of the power lattice. While this strategy has accomplished good results in the environment of group-MUS for Horn formulae, we could not confirm this for our problem domain. Yet MUS extraction based on an incremental LTUR can produce slightly better runtime data than QuickXplain. However, we have yet to examine in detail other improvements that can be made based on XPLORER, e.g., to exclude all subsets of an MUS by not only ignoring all of its supersets, but also adding a clause for blocking assumptions further down in the lattice [Mal13]. As every subset of a MUS is satisfiable by definition, this technique focuses the search even more towards the extraction of conflicts. In light of the results, we plan on further investigating strategies from infeasibility analysis to improve upon the computation of abductive diagnoses in the context of Horn formulae. In particular, it might be of interest to considering MUS enumeration methods in conjunction with the restriction of the search space as performed by HS-DAG to compute diagnoses very efficiently. Considering conflict extraction approaches, such as MergeXplain, that not only derive a single refutation but compute several at once is a possible area for future research.

Exploiting Structural Metrics in Abductive Diagnosis

6

> *The most exciting phrase to hear in science, the one that heralds the most discoveries, is not 'Eureka!' (I found it!) but 'That's funny...'*
>
> **— Isaac Asimov**
> *Ascribed to Isaac Asimov by the program "fortune"*. 1987.

This chapter is based on the following publications:

- [KW16a] Roxane Koitz and Franz Wotawa. „Improving Abductive Diagnosis Through Structural Features: A Meta-Approach". In: *Proceedings of the 2016 International Workshop on Defeasible and Ampliative Reasoning*. 2016, pp. 1–9
- [KW16c] Roxane Koitz and Franz Wotawa. „On Structural Properties to Improve FMEA-Based Abductive Diagnosis". In: *Proceedings of the Workshop on Knowledge-based Techniques for Problem Solving and Reasoning*. 2016, pp. 1–7
- [KHW18] Roxane Koitz-Hristov and Franz Wotawa. „Applying Algorithm Selection to Abductive Diagnostic Reasoning". In: *Applied Intelligence* (2018). ISSN: 1573-7497. URL: https://doi.org/10.1007/s10489-018-1171-9

The metrics for bijunctive definite Horn theories have first been published in [KW16c] with minor adaptations in [KW16a]. For our experiments, we rely on the evaluations published in [KW16c] and [KW16a]. In the second portion of the chapter, we investigate Horn abduction. This research has been published in [KHW18].

6.1 Motivation

The previous two chapters have investigated mechanisms for solving model-based diagnosis problems and in particular, we have examined (1) models equivalent to the simple set-cover theory resulting from the automatic model transformation based on FMEAs and (2) a more general view on Horn theories. From the experimental portion of the chapters, we could see that even though there are certain approaches more suitable for a certain system description category, the samples "simple" to one algorithm are not necessarily efficiently solved by all other procedures used in the comparison. As we know abduction generally is an intractable problem, which grows exponentially in the size of the model, though there exist certain subsets of logics where abduction is tractable. There are practical examples of NP-complete problems that can be solved quite efficiently, however, usually there is no universally "best" algorithm operating well on all problem instances [Kot14]. Choosing the computation method in regard to the particular example at hand can provide better performance results [Kot14]. Hence, within this chapter we investigate algorithm selection as a means to efficiently compute abductive explanations in the context of diagnosis. First formalized by Rice [Ric76], algorithm selection aims at identifying the "best performing" approach for a specific problem instance. The basic building blocks within this framework are (1) a portfolio of algorithms to choose from, (2) empirical performance data of the algorithms on representative examples, and (3) a set of features, which are used to get a

notion of the difficulty of a problem case [Hut+06]. On grounds of the empirical data and the feature vector, a predictor is trained that is capable of determining the most suitable approach for a distinct sample from the problem space. Machine learning has been identified as a feasible technique for building a prediction tool. Leyton-Brown et al. [LB+03] describe their portfolio approach to algorithm selection, where they train an empirical hardness model for each algorithm within their portfolio to forecast each approach's computation time on the instance and execute the one predicted as most efficient. While there is research on manually creating models for the predictor [Wei+08], most of the time a machine learning classifier is exploited that categorizes a new problem instance at hand as one of the methods in the portfolio. Static portfolio approaches have a finite and fixed set of algorithms to chose from (though there exist methods where the number of algorithms in the portfolio is based on the training data), while a dynamic portfolio is built online and may be constructed of algorithmic blocks [Kot14]. Generally, since the seminal paper by Rice [Ric76] various strategies have been proposed in the field of algorithm selection, e.g., in regard to the selection itself (e.g., a single algorithm is used to solve the entire problem or interleaved/parallel executions of various approaches are applied to a single problem instance), the number of models built (e.g., a single one predicts the "best" method or one model per approach in the portfolio is exploited), or the time of choosing the algorithm (e.g., at the beginning or several times during solution search). The interested reader is referred to Kotthoff [Kot14], who presents a comprehensive overview of different algorithm selection techniques. In addition, the literature provides several successful applications of algorithm selection, for instance, in the domain of SAT [Xu+08].

At the intersection of abduction and algorithm selection, there is the work by Guo and Hsu [GH07]. The authors propose algorithm selection in the context of deriving the MPE in probabilistic inference. Differencing from other work on algorithm selection, the authors first use classification to determine whether the problem is solvable and then either exploit clique-tree propagation to derive the exact solution or apply a second classifier to identify the "best" approximation procedure. While not directly usable for abductive reasoning, Malitsky et al. [Mal+14] apply the portfolio approach to enumerate MCSes, which can be exploited to derive explanations based on their hitting set duals as we have discussed in Section 4.2. Instead of choosing a single solver for the current problem instance, the authors use a technique that switches between various enumeration procedures multiple times. After a fixed timeout the next solver is nominated to compute the remaining solutions. The decision for the proceeding solver is derived on demand and to ensure already computed solutions are not extracted again, blocking clauses are added whenever a new enumeration procedure is chosen.

In this chapter, we first restrict the problem space to propositional Horn clause models which can be automatically generated from FMEAs available in practice. As discussed previously, the resulting logical system descriptions are characterized by certain structural properties. We utilize these characteristics as features for the algorithm selection process. Based on the empirical evaluation in Section 4.3.2, we extract the attributes for a collection of instances. On basis of the performance data and the features we train a machine learning classifier to forecast the algorithm most suitable in regard to its runtime for a particular abductive diagnosis scenario. We embed the selection process within a meta-algorithm that generates the structural metrics for a given diagnosis problem, categorizes it on the previously trained classifier, and computes the diagnoses using the algorithm chosen by the predictor. In a second analysis, we rely on the evaluation of Chapter 5. Here, we slightly adapt the set of attributes used to train the predictor for Horn models.

This chapter is structured as follows; in the next section, we discuss our meta-approach, while in Section 6.3 we present the structural characteristics forming our features for

the algorithm selection in case of the simple bipartite abduction problems. Subsequently, we empirically evaluate our meta-approach in comparison to consistently using the same technique in our portfolio. In the second portion starting from Section 6.4, we adapt the features slightly in order to represent Horn clause theories rather than simple implicative Horn clauses. Again, we show empirical results contrasting algorithm selection to a single-method strategy.

6.2 Meta-Approach

Algorithm selection aims at identifying the most appropriate method out of a portfolio of techniques for a given problem instance in regard to its performance. Performance in this context is most commonly associated with the computation time but could also refer to accuracy or simplicity. The model as described by Rice [Ric76] advocates for the use of features inherent to the problems within the problem space in order to accurately map a new sample to its most effective or efficient algorithm. This mapping is based on empirical performance data on representative samples of the approaches present in the algorithm space. On basis of the features extracted and the execution records, a predictor is trained that can determine aspects of the problem influencing the performance of a technique. Thereby each problem is described by a set of attributes that together with execution data allows a predictor to forecast the most valuable algorithm on an instance. Machine learning has been identified as a feasible approach to use as a prediction tool [Kot14].

Generally, there are two main variants; either one algorithm of the portfolio is to be selected based on a single predictor or for each approach within the portfolio the performance metric should be determined as a basis of the selection. The latter requires a distinct empirical hardness model for each method within the portfolio and thus whenever the algorithm for a new instance is to be selected, for each approach a prediction has to be made [Hut+06; Kot14]. SATzilla [Xu+08] is an example of such a portfolio approach within the domain of SAT solvers. For our meta-technique, however, we consider the first alternative, where we train a single classifier for all abductive reasoning methods to select a single approach for execution.

We consider a 1-of n portfolio with n algorithms to choose from but only one is selected and executed to solve the diagnosis problem [Xu+08]. Within the context of troubleshooting our meta-approach operates the following way: as mentioned the foundation of model-based diagnosis is a description of the system under consideration. Thus, we ensure that the majority of the features can be computed offline on the diagnosis models present. Further, within the offline phase the empirical data on computation times of the various abductive reasoning approaches can be collected and on basis of the metrics and the runtime information a machine learning classifier *CL* is trained.

Algorithm METAB describes the online portion of the meta-approach, which is executed whenever new diagnoses are to be computed. The diagnosis process is triggered by a detected anomaly; at this time we retrieve our previously learned machine learning classifier *CL* as well as the offline determined metrics $\phi_{offline}$ of the diagnosis model. Online we have to collect the current *PHCAP*'s instance-based features ϕ_{online}, which we discuss in detail in the upcoming section. While Hutter et al. [Hut+06] state that the feature extraction method should be highly efficient, in our framework only the computation of a subset of these attributes has to be performed online, namely the computation of the metrics depending on the observations. Based on the online and offline generated attributes, we supply the feature vector ϕ with the measurements of the current diagnosis problem. By providing all features to the machine learning algorithm, we in turn retrieve a predicted best abduction method α

out of our portfolio for this specific diagnostic scenario based on the trained classifier and the instance's features. Subsequently, we can instantiate the abductive reasoning engine with the corresponding abduction technique as well as problem instance and compute the set of minimal explanations, i.e., Δ-Set.

Algorithm 6.1: METAB

Require: *A*: the set of propositional variables, *Hyp*: the set of hypotheses, *Th*: the Horn theory, *Obs*: the set of observations, *CL*: the trained classifier, $\phi_{offline}$: the previously computed model metrics
Ensure: Δ-Set: set of consistent minimal diagnoses
$\phi_{online} \leftarrow$ `computeMetrics`(*A,Hyp,Th,Obs*) ▷ Computes the instance-based features
$\phi = \phi_{offline} \cup \phi_{online}$ ▷ Combine the attributes to the entire feature vector
$\alpha \leftarrow$ `predict`(ϕ, CL) ▷ Forecasts the best performing algorithm α for the diagnosis problem based on the features ϕ and the classifier *CL*
Δ-Set $\leftarrow$ `diagnose`$(\alpha,$*A,Hyp,Th,Obs*) ▷ Computes solutions with α for the diagnosis problem
return Δ-Set

6.3 Bipartite Models

In Chapter 4, we have described possibilities to solve abduction problems based on an automatic compilation from FMEA documents. In particular, we uncovered that the models are rather simple in the sense that they can be mapped to the simple parsimonious set covering theory, i.e., the result of the mapping is a model consisting of bijunctive definite Horn clauses. Thus, we can easily define their structural properties based on various graph representations. In the simplest case, the theory is characterized as a DAG $G = (V, E)$. Each proposition in A corresponds to a vertex, while the edges between nodes are determined by connections within the theory. For each clause c, there is a directed edge leading from the node representing the negative literal in the clause to the node of the positive literal, i.e., $E = \{x_i \rightarrow x_j | c \in Th, x_i \textit{ is the negative literal in } c \textit{ and } x_j \textit{ is the positive literal in } c\}$ [EM03]. In this way, the DAG simply is the dependency graph of the Horn theory. An additional feature of G is that it contains two disjunctive node sets, namely the propositional variables constituting the causes and the effects, respectively, i.e., we can depict the theory as a bipartite graph. This characterization is equivalent to the associative network as described by Peng and Reggia [PR90] in their set covering approach.

As stated the models we are considering are bijunctive definite Horn clauses. Given a model of this type we can easily represent it as a hypergraph $H = (V, E)$, where V is the set of vertices and E constitutes the set of hyperedges. The nodes of the hypergraph represent the propositional variables, while the hyperedges are determined by the theory. For each clause there exists a hyperedge containing all propositional variables of said clause, i.e., $\forall a \in A \rightarrow a \in V$ and $\forall c \in Th \rightarrow \bigcup_{l \in c} |l| \in E$ where $||$ is a function mapping literals to the underlying propositions ignoring negations, i.e., $|\neg p| = p$ and $|p| = p$ for all $p \in A$. Following this representation we can assign a label to each vertex within a hyperedge E, such that:

$$label(v) = \begin{cases} \{v\} & \text{if } v \in Hyp \\ \bigcup\limits_{x \in E \wedge x \neq v} label(x) & \text{otherwise} \end{cases}$$

In case a vertex represents a manifestation, its label correspond to its *causes*-set, as it holds the hypotheses responsible for the effect. Note here that our label differentiates from an ATMS label, where each label is a set of environments and each environment again is represented by a set of assumptions. We can utilize the labels of the nodes corresponding to the observations to compute the abductive diagnoses as hitting sets. Note that by relying on this notion we could further handle intermediate effects, which are not contained in the simple bipartite problems, however, can occur in more expressive languages.

6.3.1 Structural Metrics

Considering the graph representations of the model, we can extract certain characteristics of their structure that we subsequently use within the algorithm selection process. The most intuitive measure of complexity is the cardinality of *Hyp* as abductive diagnosis is possibly exponential in the number of causes to consider. In addition, we collect the number of effects and connections within the theory.

Outdegree and Indegree

Based on the DAG, we can compute for each vertice representing a hypothesis its outdegree, which specifies the number of manifestations affected by said cause. Similarly, we measure the indegree of each effect, i.e., the number of hypotheses inferring the manifestation. Considering the set covering framework, we can define the degrees as follows:

$$outdegree(p_i) = |effects(p_i)|$$

$$indegree(p_j) = |causes(p_j)|$$

Collected over the entire model these measures provide an intuitive metric of the basic magnitude and connectedness of the theory.

Covering and Overlap

Several disorders may cover the same effect, i.e., a manifestation can be explained by multiple causes. On basis of this we can define a covering metric for each pair of hypotheses as the ratio between the number of common effects and the total number of symptoms induced by the hypotheses:

$$covering(p_i, p_j) = \frac{|effects(p_i) \cap effects(p_j)|}{|effects(p_i) \cup effects(p_j)|}$$

In a similar manner, we define the overlap of two effects as their common sources in relation to all their causes:

$$overlap(p_i, p_j) = \frac{|causes(p_i) \cap causes(p_j)|}{|causes(p_i) \cup causes(p_j)|}$$

Peng and Reggia [PR90] define a pathognomonic effect as an observation with a single cause. Thus, whenever a pathognomonic symptom is involved, we do not compute an overlap relation. By collecting these measures for any pair of hypotheses or effects, we can compute a value over the entire model.

Independent Diagnosis Subproblem

Whenever there exist several subproblems in our theory we refer to them as independent diagnosis subproblems. If several subproblems exist, the DAG and all other graphs representing the model are disconnected and each independent diagnosis subproblem itself is a

connected subgraph. In case all effects are pathognomonic, then each cause-effect relation represents its own independent diagnosis subproblem and thus we can observe that the model is orthogonal. As an additional measure to the number of subproblems we further compute the average size, i.e., number of nodes involved, over all independent diagnosis subproblems in case several exist.

Path Length

Another measure of connectedness within the model is the minimal path length between any two nodes on the hypergraph. In particular, we measure the length of the minimal path between nodes representing hypotheses, thus we compute the minimal number of hyperedges to be traversed between each pair of hypothesis vertices. Note that for a single model there are possibly several hypergraphs depending on the number of independent diagnosis subproblems, thus we disregard paths between nodes belonging to different subproblems.

Local Clustering Coefficient

The clustering coefficient is a known measure of node clusters within a graph. It is evident that we cannot compute a clustering coefficient from the graph representations used so far, i.e. the DAG, bipartite graph and hypergraph, in case we want to account for the two disjoint node classes. Therefore, we transform the bipartite graph by projection. In particular, we remove all nodes corresponding to manifestations and link two cause vertices v_{h_i} and v_{h_k} whenever they imply the same effect, i.e., $\mathit{effects}(h_i) \cap \mathit{effects}(h_k) \neq \emptyset$. Based on the resulting undirected graph featuring only the nodes corresponding to hypotheses, we compute for each node the local clustering coefficient as $c = \frac{2n}{k_i(k_i-1)}$, where n is the number of neighbors of the node and k_i the number of edges between the neighbors of n[1]. While in network analysis the projection of bipartite graphs results in coefficients differentiating from typical one-mode networks, this does not pose an issue in our case as we are solely interested in this feature as an attribute to our meta-algorithm. Thus, the clustering coefficient provides for our models another measure of covering between hypotheses.

Kolmogorov Complexity

A simple encoding-based measure on a graph is its Kolmogorov complexity, which defines a value equal to the length of the word necessary to encode the graph. A straightforward manner in this context is to compute the complexity based on String representation of the adjacency matrix. For this, we create a undirected graph by replacing each directed edges with an indirect edge and use am approximation of the Kolmogorov complexity[2].

Observation Dependent Metrics

Since not only the topology of the model is of interest, but also the structure of the current diagnosis problem, we measure the indegree and the overlap among the elements of *Obs* as well as the number of diagnosis subproblems involving variables of *Obs*, in case several exist.

[1] Clustering coefficients, who's value are not defined (division by zero) are considered as 0 in the averaging over the entire model.

[2] `https://goo.gl/8Ei6zs`

Example 6.1

Consider the following *KB*:

$$A = \{h_1, h_2, h_3, o_1, o_2, o_3\}, Hyp = \{h_1, h_2, h_3\},$$

$$Th = \{\ h_1 \rightarrow o_1, h_2 \rightarrow o_1, h_2 \rightarrow o_2, h_3 \rightarrow o_2, h_3 \rightarrow o_3\ \}$$

We see that there are three hypotheses, three manifestations and the theory consists of five bijunctive definite Horn clauses. Based on the logical model, we can construct the graphs and collect, for instance, the following data on the model:

Figure 6.1: DAG and hypergraph representation. The DAG shows shared hypotheses (left oval) and common effects (right oval) for pairs of nodes.

- *Outdegree and Indegree*: In regard to the example, we can observe $outdegree(h_2) = |effects(h_2)| = 2$ and $indegree(o_3) = |causes(o_3)| = 1$.
- *Covering and Overlap*: Figure 6.1 depicts on the right hand side of the DAG the shared observation o_2 between h_2 and h_3 as a blue oval. Thus, we can see that $covering(h_2, h_3) = \frac{1}{3}$. The overlap of o_1 and o_2 at h_2 is shown as a magenta colored oval on the left side of the DAG in Figure 6.1 ($overlap(o_1, o_2) = \frac{1}{3}$).
- *Independent Diagnosis Subproblem*: Imagine the clause $h_2 \rightarrow o_2$ missing from the theory. In this case, we would have two independent diagnosis subproblems, namely one including h_1, h_2 and o_1 and the other one consisting of h_3, o_2 and o_3. Hence, the average size of the diagnosis problems is 3.
- *Path Length*: Considering the hypergraph in Figure 6.1, we can observe $path(h_1, h_2) = 2$.
- *Clustering Coefficient*: In this example, the projection on the bipartite graph based on the theory constructs the undirected graph with nodes h_1,h_2, and h_3 and the undirected edges $\langle h_1, h_2 \rangle$ and $\langle h_2, h_3 \rangle$. Each node's clustering coefficient computation leads to 0 (due to division by zero), hence, the clustering coefficient of the entire model is 0.
- *Kolmogorov Complexity*: For this example, we can obtain a Kolmogorov Complexity of 14.

Assuming o_2 and o_3 are being observed, i.e., $Obs = \{o_2, o_3\}$, then $label(o_2) = \{h_2, h_3\}$ and $label(o_3) = \{h_3\}$. Computing the minimal hitting set based on the two labels, we retrieve h_3 as our abductive explanation.

- *Observation Dependent Metrics*: We can extract the overlap information based on o_2 and o_3, i.e., $overlap(o_2, o_3) = \frac{1}{2}$. Considering again that the clause $h_2 \rightarrow o_2$ is not encompassed in *Th*, the second diagnosis subproblem consisting of h_3, o_2 and o_3 would comprise both observations.

Table 6.2 lists the set of features considered in the meta-approach. Considering the requirement to compute features efficiently within an algorithm selection set-up, let us

1. Logic Model Specific:
- *Number of hypotheses*
- *Number of effects*
- *Number of causal relations*, i.e., clauses in the theory

2. Directed Graph:
- *Outdegree* of hypothesis nodes: maximum, average, standard deviation
- *Indegree* of effect nodes: maximum, average, standard deviation
- *Covering*: maximum, average, standard deviation
- *Overlap*: maximum, average, standard deviation
- *Number of independent diagnosis subproblems*
- *Average size of independent diagnosis subproblems*

3. Undirected Graphs:
- *Local clustering coefficient* (maximum, average, standard deviation)[a]
- *Kolmogorov complexity* based on adjacency matrix [b]

4. Hypergraph:
- *Path length* (maximum, average, standard deviation)[c]

5. Instance Specific/Observation Dependent:
- *Number of observations*
- *Indegree current observation nodes*: maximum, average, standard deviation
- *Overlap current observation*: maximum, average, standard deviation
- *Number of independent diagnosis subproblems* including current observations

[a] Based on the projection of the bipartite graph only containing hypothesis nodes.
[b] Based on an undirected graph constructed from the DAG.
[c] Path length between hypothesis vertices.

Figure 6.2: Features.

define which portions of the meta-algorithm have to be actually derived online. Creating the diagnosis models, computing the basic model features, i.e. feature sets *1* to *4*, and training of the classifier are all performed offline. Thus, online only the observation specific metrics are extracted, i.e., feature set *5*, the algorithm appropriate for the problem instance is predicted, and the diagnoses are calculated.

6.3.2 Empirical Results

In this portion, we present empirical results of our meta-approach in comparison to always choosing a single method from the portfolio of algorithms. The benchmarks and empirical performance data we are using stem from the evaluations of the simple diagnosis problems in Section 4.3.2. We employ a 1-of 5 portfolio for both experiments, i.e., we select one approach from the static algorithm space containing five methods which can be utilized for abductive reasoning based on bipartite diagnosis problems. As with the first evaluation in Section 4.3.2, we use an ATMS as a general abduction engine for propositional Horn clauses. Besides the ATMS our portfolio holds various hitting set algorithms, which are capable of computing minimal diagnoses as shown in Chapter 4. The hitting set routines we include in our meta-approach are the following: BHS-Tree, HS-DAG, HST, and the Boolean approach. As we have seen in the first evaluation that the Boolean algorithm has been the most efficient contester, in the second assessment of in Section 4.3.2 we exchange the Boolean algorithm for Berge's algorithm in order to be used in an on the fly hitting set computation. MetAB itself is implemented in Java and for the machine learning part of our meta-algorithm we employ the Waikato Environment for Knowledge Analysis (WEKA) library [Hal+09], which provides a vast number of classification methods.

Superiority of the Boolean Algorithm

To collect runtime data for the training and test set for classification, we exploit a Java implementation of an ATMS as well BHS-Tree[3] and a Python library [QP14][4] of the remaining

[3] `http://www.ist.tugraz.at/modremas/index.html`
[4] `http://modiaforted.ist.tugraz.at/downloads/pymbd.zip`

	Artificial Examples	
Classification Method	Multinomial Logistic Regression	Multilayer Perceptron
|Training Set|	210	1696
|Test Set |	42	424
Total Test Time	$<$ 1 ms	5 ms
Correctly Classified Instances	30 (71.43 %)	308 (72.64 %)
Incorrectly Classified Instances	12 (28.57 %)	116 (27.36 %)
Mean absolute error	0.17	0.17

Table 6.1: Classification Statistics.

hitting set methods. With each experiment run we collected the metrics listed in Figure 6.2 except the Kolmogorov complexity and the average size of the independent diagnosis subproblems that we will use only in the second experiment. All these metrics build our feature vector for the classification. The variable to be predicted is the algorithm, i.e., *ATMS*, *BHS-Tree*, *HS-DAG*, *HST*, or *Boolean*, which would be the most efficient on the current diagnosis problem. Each experiment data series was split into a training set comprising 80 % of the data and a test set of 20 %. Before selecting the classification method, we performed cross validation on several classification algorithms available in WEKA on the training data. Based on the accuracy obtained we decided to use a multilayer perceptron as the classifier for the FMEA-based models and multinomial logistic regression for the artificial examples.

As can be seen in Table 6.1 the classification based on the metrics reaches a satisfactory success rate on the FMEA-based as well as artificial examples. The confusion matrices in Table 6.2 and Table 6.3 show the number of correctly and wrongly classified instances. The rows represent the actual number of instances, i.e., the number of samples the corresponding algorithm was the most efficient, while the columns show the predicted outcome. For example in Table 6.2, the cell in the first row in the first column states that on the artificial examples *Boolean* was correctly predicted 20 times. Moving three rows down, we see that two times the classifier predicted the *Boolean* algorithm to be the most efficient procedure when in fact *BHS-Tree* was the fastest. Moreover, four examples were wrongly classified as *BHS-Tree* instead of the *Boolean* approach. Note that for the FMEA samples *HST* and *HS-DAG* were never predicted or actually measured to be the best performing algorithm, the same holds for the *ATMS* in the artificial examples. Further, we see that the *Boolean* approach was the most efficient, followed by *BHS-Tree* on both test sets. From the confusion matrix we also infer that the neural network had difficulties in classifying instances where the *ATMS* succeeded, as it categorizes several instances incorrectly and the same holds for the *HS-DAG* in the artificial examples. From a deeper analysis of the results, yet, we know that whenever the classifier categorizes the problem incorrectly, the suggested algorithm is the second or third most efficient.

To discover whether our meta-approach provides an efficiency improvement, we compared computation time on both test sets for all methods, i.e., our meta-algorithm and each abductive reasoning technique. The runtime for the meta-algorithm is determined by (1) the computation of the metrics, (2) the time it takes to create the feature vector, supply it to the classifier, and predict the best algorithm, and (3) the diagnosis time of the suggested procedure. Figure 6.3 shows the cumulative log runtime for the test data for growing computation time. We observe that on the artificial examples, the meta-approach performs better than on the FMEA sample. The plots in Figure 6.4 depict the statistical information on the different abduction approaches. As can be seen from the figure, our meta-algorithm is not able to outperform on average (2.22 ms) all direct diagnosis methods (*Boolean* 0.48 ms, *BHS-Tree* 1.36 ms, *HS-DAG* 1.09 ms, *ATMS* 3.14 ms, *HST* 2643.17 ms) for the FMEA-based examples. The reason being that the mean time to collect the metrics of the *PHCAP* is close the actual diagnosis time of these problems. In case of the artificial examples, our approach

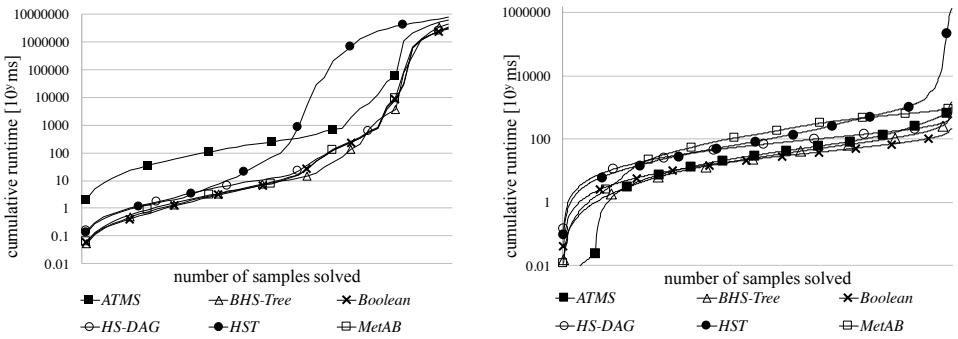
		Predicted					
		Boolean	*ATMS*	*BHS-Tree*	*HST*	*HS-DAG*	**Total**
Actual	*Boolean*	20	0	4	1	1	26
	ATMS	0	0	0	0	0	0
	BHS-Tree	2	0	6	0	0	8
	HST	0	0	1	0	0	1
	HS-DAG	3	0	0	0	4	7
	Total	25	0	11	1	5	42

Table 6.2: Confusion matrix of the Artificial benchmark.

		Predicted					
		Boolean	*ATMS*	*BHS-Tree*	*HST*	*HS-DAG*	**Total**
Actual	*Boolean*	186	22	25	0	0	233
	ATMS	35	32	12	0	0	79
	BHS-Tree	16	6	90	0	0	112
	HST	0	00	0	0	0	0
	HS-DAG	0	0	0	0	0	0
	Total	237	60	127	0	0	424

Table 6.3: Confusion matrix of the FMEA benchmark.

performs well. On average we can observe the following runtimes: *Boolean* 74,752.23 ms, *BHS-Tree* 103,065.19 ms, *HS-DAG* 82,008.41 ms, *ATMS* 144,276.18 ms, *HST* 180,849.51 ms, and *MetAB* 71,812.24 ms. This is due to the fact that on these instances the computation of the properties only demands a fraction of the actual diagnosis run time as the models are larger in general.

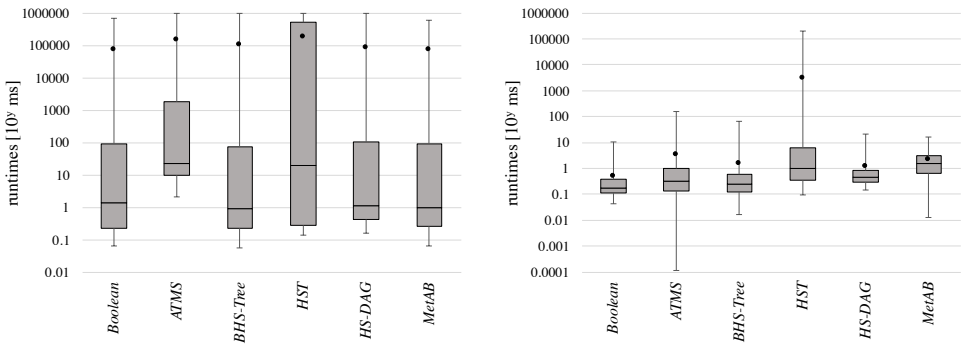


(a) Artificial Examples. **(b)** FMEA Examples.

Figure 6.3: Cumulative runtime for the test sets.

Computing Abductive Diagnoses On The Fly

In this evaluation, we disregard the Boolean approach as a hitting set technique and replace it with Berge's algorithm due to its capabilities of deriving diagnoses on the fly. The empirical performance data stem from the second experiment described in Section 4.3.2 that are based on artificial diagnosis problems. Before selecting the classification method, we performed cross validation on several classification algorithms available in WEKA on the training data. Based on the accuracy obtained we decided to use WEKA's implementation of a general algorithm for locally weighted learning *LWL*. While within this experiment we did not compare different classifiers besides the initial informal evaluation, we do believe that examining various machine learning algorithms will be of interest in future research on this matter. *LWL* performs prediction by building a local model based on the weighted neighborhood data of the attribute of interest. In our case, this attribute is nominal and simply corresponds to the algorithms name. In regard to the parameters, we utilized a



(a) Artificial Examples. **(b)** FMEA Examples.

Figure 6.4: Statistical distribution for the runtimes [10^y ms] on the test set.

Table 6.4: Confusion Matrix for the artificial test set. The rows represent the actual number of instances within the category, while the columns show the predicted outcome.

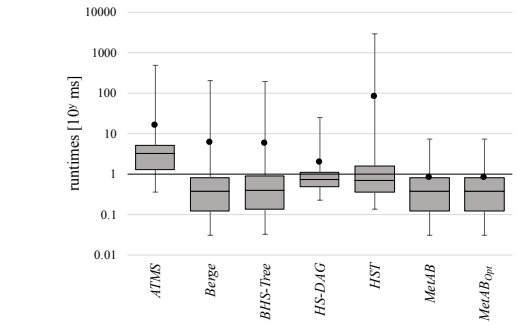
		Predicted					
		Berge	*HS-DAG*	*ATMS*	*BHS-Tree*	*HST*	**Total**
Actual	*Berge*	27	0	0	0	0	27
	HS-DAG	1	0	0	0	0	0
	ATMS	2	0	0	0	0	2
	BHS-Tree	8	0	0	0	0	8
	HST	0	0	0	0	1	1
	Total	38	0	0	0	1	39

brute force nearest neighbor search algorithm, included all neighbors in the linear weighting function, and chose an entropy based classifier.

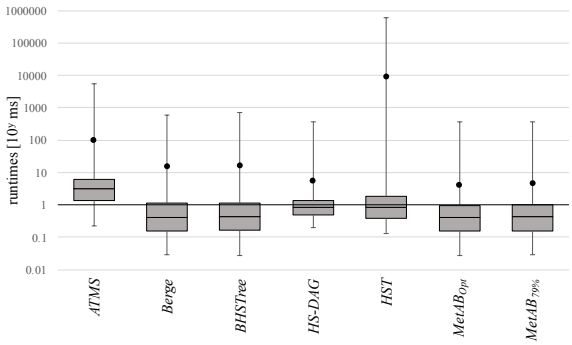
The classification utilizing *LWL* and based on the metrics listed in Figure 6.2 reaches a satisfactory success rate of 71.79 % (Mean absolute error=0.22, Root mean square error=0.31) correctly predicted instances, i.e., the selected algorithm was in fact the most efficient on the problem. The confusion matrix in Table 6.4 shows the number of correctly and wrongly classified instances. From the contingency table it is apparent that within the test set *Berge*'s algorithm was the dominant approach, thus, our predictor favors the method and classifies all but one instance as *Berge*'s algorithm. A known limitation of this type of algorithm selection, where simply a single approach is chosen and executed on the instance, is that in case the prediction is incorrect the meta-approach might be rather inefficient. Note that in case our classifier chose a slower approach, the selected algorithm was the second fastest within the portfolio. On the test set the meta-approach was on average 1.57 % slower than an optimal algorithm selection ($MetAB_{Opt}$), i.e., the predictor would classify every instance correctly, would have been. Figure 6.5a shows the distribution of the log runtime data for the test set.

We explored WEKA's attribute selection in order to determine whether we could remove certain features while achieving the same prediction accuracy. Utilizing the meta-classifier with the *LWL* classifier, we examined various attribute selection approaches on the training data and could diminish the set of features significantly from 32 to four[5]. The number and composition of the reduced attribute set depends highly on the performed selection process. For example, attribute selection on grounds of the information gain results in a different feature set: the number of current diagnosis problems, the number of observations, the average path length on the hypergraph and its standard deviation. Utilizing the support

[5]The feature to predict is not considered in this number and of course remains within the set of features after attribute selection.



(a) Test set.



(b) Entire sample set.

Figure 6.5: Underlying statistical distributions of the log runtimes.

vector machine-based reduction, we receive the number of observations, the standard deviation of the indegree of the nodes representing the observations, the average current covering relation and its standard deviation as well as as the average of the covering relation over the entire model. As can be seen the size of *Obs* plays an essential role in predicting the preferable algorithm. In regard to the remaining selected properties they provide information on the *PHCAP*, i.e., the current observations, and various metrics on how hypotheses are connected through effects.

A premature analysis of the results of the test data would suggest that applying *Berge*'s method to every instance would yield the optimal runtime for most problems. However, from Figure 6.5b, where have depicted the distribution of the log runtimes of the various approaches, we deduce that based on the entire set of problems, i.e., test and training, *Berge* is not the most efficient approach as on several instances its computation time is notably larger than of other algorithms. On the entire sample only considering the algorithms from the portfolio, we observe that *HS-DAG* is on average the best performing approach followed by *Berge*'s algorithm and *BHS-Tree*, while the still outperforms *HST*. Furthermore, based on the prediction accuracy on the test set, we have created a mock meta-approach ($MetAB_{71.79\%}$) with 71.79 % accuracy as we have experienced on the test data. Thus, in 71.79 % of the samples we record for this approach the optimal time and for the remaining 28.21 % the second fastest time. We choose the instances with the slower runtimes randomly and picked the second most efficient algorithm. This mock meta-approach would still outperform, the other algorithms in the portfolio.

6.4 Horn Models

The second focus of our research has been the evaluation abduction methods suitable for more expressive representations. In particular, we exploit the empirical performance data we have obtained from the evaluation in the previous chapter. However, since the models' characteristics differ from the simple bipartite system descriptions, we adapt the set of features slightly in order to account for these changed circumstances.

6.4.1 Structural Metrics

While we can rely mostly on the previously defined metrics, there are some adaptations necessary. First, we can construct in the same manner as before a directed graph G[6]. In addition, we have discussed a hypergraph representation to propagate the label information holding the hypotheses preceding a certain node. First, we cannot use the label information for deriving abductive explanations as in the previous chapter, since the Horn abduction problem is not equivalent to the set covering problem. However, we can utilize the label information as an additional structural feature. Although the hypergraph representation is still applicable on Horn clauses, we can also achieve the same label properties based on the directed graph by redefining how a node's label is computed. For each vertex $v \in G$, the label contains all hypotheses said node is (directly and indirectly) caused by:

$$\mathit{label}(v) = \begin{cases} \{v\} & \text{if } v \in \mathit{Hyp} \\ \bigcup\limits_{(x,v)\in E} \mathit{label}(x) & \text{otherwise} \end{cases}$$

Based on the directed graph and the computed label for each vertex v, which contains all hypotheses said node is (directly and indirectly) caused by, we can state an additional covering and overlap metric. Note that we do not contemplate the connectives of the variables, hence, a label in our case is a simple set. Given the definition of a label, we define an overlap based on labels:

$$\mathit{overlap}_{\mathit{label}}(p_i, p_j) = \frac{|\mathit{label}(p_i) \cap \mathit{label}(p_j)|}{|\mathit{label}(p_i) \cup \mathit{label}(p_j)|}$$

While *overlap* only takes into account the direct causes of a proposition, $\mathit{overlap}_{\mathit{label}}$ uses the information of all predecessors of a node. By collecting these measures for any pair of hypotheses or effects, we can compute a value over the entire model.

Further, for this type of model we disregard the clustering coefficient, however, we create an undirected graph based on the directed graph, such that it only consists of nodes that cause another node, i.e., we only consider propositions as nodes, which are causing other propositions. An edge is created between two propositions, in case they are directly causing the same effect. In particular, we measure the length of the minimal path between these nodes. Figure 6.8 lists all features considered for this type of model.

Example 6.2

For describing the features, we will use the Horn clause model with $\mathit{Th} = \{e_1 \wedge H_1 \rightarrow e_2, e_1 \wedge H_1 \rightarrow e_3, H_2 \rightarrow e_3, H_2 \rightarrow e_4, e_3 \rightarrow e_4, H_3 \rightarrow e_5\}$ as an example. Fig. 6.6 shows G created on basis of the model. The labels are as follows:

[6] That is equivalent to the dependency graph of the Horn theory.

v	H_1	H_2	H_3	e_1	e_2	e_3	e_4	e_5
$label(v)$	H_1	H_2	H_3	$\emptyset$	H_1	H_1, H_2	H_1, H_2	H_3

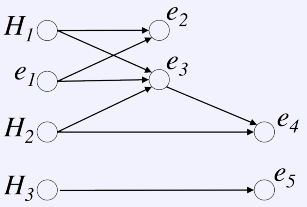


Figure 6.6: Directed graph G.

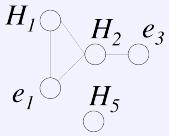


Figure 6.7: Undirect graph of cause nodes.

For instance, we can extract $outdegree(H_1) = 2, indegree(e_2) = 2, covering(H_1, H_2) = \frac{1}{3}, overlap(e_3, e_4) = \frac{1}{4}$ and $overlap_{label}(e_3, e_4) = 1$. We also have two independent diagnosis subproblems, namely one including H_1, H_2, e_1, e_2, e_3 and e_4 and the other one consisting of H_3 and e_5. To compute the path length between nodes functioning as causes, we construct the graph in Figure 6.7. Based on the undirected graph, we can see for instance that $path(H_1, e_3) = 2$.

Logic model specific:
1. *Number of hypotheses*
2. *Number of effects*, i.e., propositions in $A \setminus Hyp$
3. *Number of causal relations*, i.e., clauses in the theory

Directed Graph:
4-6. *Outdegree* of hypothesis nodes: maximum, average, standard deviation
7-9. *Indegree* of effect nodes: maximum, average, standard deviation
10-12. *Covering*: maximum, average, standard deviation
13-15. *Overlap*: maximum, average, standard deviation
16-17. *Overlap with labels*: maximum, average, standard deviation
18. *Number of independent diagnosis subproblems*
19. *Average size of independent diagnosis subproblems*

Undirected Graph:
20. *Shortest path between causes*: maximum, average, standard deviation
21. *Kolmogorov complexity* based on adjacency matrix

Instance specific/Observation dependent:
22. *Number of observations*
23-25. *Indegree current observation nodes*: maximum, average, standard deviation
26-28. *Overlap current observation*: maximum, average, standard deviation
29-31. *Overlap with labels current observation*: maximum, average, standard deviation
32. *Number of independent diagnosis subproblems* including current observations

Figure 6.8: Features.

6.4.2 Evaluation

In this section, we evaluate the feasibility of our meta-approach based on the an empirical study we have conducted in the previous chapter. This foregoing evaluation has comparatively analyzed conflict-driven and consequence finding techniques for abductive reasoning based on two benchmarks; one set of experiments utilizes examples stemming from practice, while the other encompasses artificially generated diagnosis problems. The objectives of the evaluation in this paper are two-fold; on the one hand, we aim at assessing the quality of the structural attributes to train a machine learning classifier to forecast the most efficient algorithm in regard to its runtime for a specific *PHCAP* instance. On the other hand, we want to determine the overall efficiency of the meta-approach in comparison to consistently using a single abductive diagnosis techniques in the portfolio.

The portfolio of algorithms we utilize in this evaluation is comprised of the following methods: abduction with the ATMS (*ATMS*), abduction as consequence finding via SOL-resolution (*CF*), conflict-driven search via HS-DAG (*HS-DAG* minimizes the solutions at the end of the computation, while *HS-DAG*$_{QX}$ minimizes each refutation right away), conflict-driven search via power set exploration (*XPLorer* utilizes the proposed MUS extraction procedure [Ari+15], while *XPLorer*$_{QX}$ exploits QuickXplain to derive minimal conflicts), and abduction under stable model semantics (*ASP*). Our meta-algorithm MetAB (*MetAB*) is a Java implementation.To create the predictor based on the features, we exploit again the WEKA library [Hal+09] which provides a vast variety of machine learning algorithms.

We exploit two sample sets for our evaluation: (1) an artificially created set of *PHCAP*s (*Artificial Samples*[7]) and (2) diagnosis problems constructed based on real system failure assessments characterizing component faults and their manifestations (*FMEA Samples*). For each *PHCAP* execution we collect the metrics listed in Figure 6.8 to gather our attributes for prediction. The feature vector itself holds one more nominal value: the class attribute. It corresponds to the algorithm's name that has solved the diagnosis problem the fastest in the experiments conducted previously. Hence, this last attribute is to be predicted. To evaluate the classification accuracy based on the features, we randomly split each benchmark into a training set comprising 80 % of the data and a test set holding the remaining 20 % of examples. Due to dividing samples arbitrarily, it is not warranted that each *PHCAP* is represented in equal measures within the test set. Hence, it may also occur that the test set comprises samples particularly suitable for a certain approach.

To determine an appropriate machine learning method, we performed model selection via 10-fold cross-validation on the training data for several classification algorithms available in WEKA. Based on the accuracy results obtained, we chose to use the random subspace method in combination with WEKA's decision tree with reduced-error pruning (REP Tree) for *Artificial Samples*. In the random subspace method, subsets of components of the feature vector are selected pseudorandomly and a decision tree is generated based solely on the chosen attributes. The resulting classifier then depends on several decision trees constructed in this manner, i.e., a decision forest. For *FMEA Samples*, we decided on WEKA's decision table majority classifier, which also relies on a suitable subset of features to build the decision table.

Table 6.5 depicts all classification statistics based on (1) the 10-fold cross-validation on the training data and (2) the evaluation on the test set. Utilizing the random subspace method we reach between 66.39 % and 67.17 % correctly labeled samples. A *PHCAP* is labeled correctly in case the predicted algorithm was the most efficient on the problem instance in our empirical performance data. For the *FMEA Samples*, we receive slightly better results with more diagnosis problems (between 76.76 % and 77.79 %) classified correctly.

Ideally, we could improve upon the accuracy results. A common strategy to enhance classification precision is to adapt the feature vector. In this respect, we explore WEKA's attribute selection in order to determine whether we could remove certain features while achieving the same or better prediction quality. Since the number and composition of the reduced feature set depends highly on the performed selection process, we conducted some informal evaluations to decide on the leading method. In the end, we chose to rank attributes in the artificial case due to their information gain in consideration of the class feature and in case of the *FMEA Samples* based on a subset of features that are statistically relevant to the class attribute (*Relief Feature Selection*). Using these methods we could diminish the set of features to fifteen and twenty-one in case of the artificial and FMEA examples, respectively[8].

[7] In the previous chapter, this set of examples was referred to as *Artificial Samples I*.
[8] Of course the attribute to predict, i.e., the most efficient algorithm, remains in the feature vector.

Table 6.5: Classification Statistics.

		Artificial Samples	FMEA Samples
	Classifier	Random Subspace Method with REP Tree	Decision Table
Cross Validation [10-fold]	|Data Set|	1660	2130
	Total Training Time [in ms]	114	604
	Total Test Time [in ms]	29	58
	Correctly Classified Instances	1102 (66.39 %)	1657 (77.79 %)
	Incorrectly Classified Instances	558 (33.61 %)	473 (22.21 %)
	Mean Absolute Error	0.17	0.16
Train and Test [80 %,20 %]	|Train|	1328	1704
	|Test|	332	426
	Total Training Time [in ms]	114	604
	Total Test Time [in ms]	29	58
	Correctly Classified Instances	223 (67.17 %)	327 (76.76 %)
	Incorrectly Classified Instances	109 (32.83 %)	99 (23.24 %)
	Mean Absolute Error	0.18	0.16

Table 6.6: Attribute Selected Classification Statistics.

		Artificial Samples	FMEA Samples
	Classifier	Random Subspace Method with REP Tree	Decision Table
	Attribute Selection Evaluator	Information Gain	Relief Feature Selection
Cross Validation [10-fold]	|Data Set|	1660	2130
	Total Training Time [in ms]	114	604
	Total Test Time [in ms]	29	58
	Correctly Classified Instances	1093 (65.84 %)	1655 (77.70 %)
	Incorrectly Classified Instances	567 (34.16 %)	475 (22.30 %)
	Mean Absolute Error	0.17	0.16
Train and Test [80 %,20 %]	|Train|	1328	1704
	|Test|	332	426
	Total Training Time [in ms]	53	650
	Total Test Time [in ms]	26	97
	Correctly Classified Instances	231 (69.58 %)	334 (78.41 %)
	Incorrectly Classified Instances	101 (30.42 %)	92 (21.59 %)
	Mean Absolute Error	0.17	0.16

As reported in Table 6.6 the results improved from 67.17 % to 69.58 % and 76.76 % to 78.41 % on the test sets. Table 6.7 lists the selected attributes according to their rank. While the suggested feature sets diverge, there are certain attributes which seemingly are important in both benchmarks, namely the metrics which characterize the basic composition of the theory, e.g., *overlap* and *indegree*.

The confusion matrices in Tables 6.8 and 6.9 give a deeper insight into the number of correctly and wrongly labeled instances of the test set based on the attribute selected classifiers. Each column of the matrix reports on the number of instances labeled a certain algorithm, while the rows represent the number of *PHCAP*s the algorithm was superior to the other approaches. For example, Table 6.8 shows that *HS-DAG* was predicted as the most efficient method 191 of 332 times, while it actually performed the best on 169 samples. Hence, on 22 samples the classifier incorrectly selected *HS-DAG*. In 33 cases, the instance was mislabeled as $HS\text{-}DAG_{QX}$ though *HS-DAG* would have been the best suited candidate. For 35 examples the situation was the other way around. This is due to both approaches experiencing a similar runtime behavior for most samples.

Table 6.10 and Table 6.11 report on the binary classification measures for each approach on the test set. The best results in each column are emphasized. Precision denotes the proportion of cases correctly labeled an approach to all the samples the algorithm was selected as the abduction method, whereas Recall is the ratio of samples correctly predicted to instances where the approach is actually the fastest. The F_1-Score is a combined metric based on the weighted average of Precision and Recall. Considering for example the *ATMS*

Table 6.7: Selected Attributes.

	Artificial Samples	*FMEA Samples*
1	*Covering*: standard deviation	*Number of observations*
2	*Overlap with labels*: standard deviation	*Overlap current observation with label*: maximum
3	*Overlap with labels*: average	*Overlap current observation with label*: standard deviation
4	*Indegree current observation nodes*: average	*Overlap current observation*: maximum
5	*Outdegree of hypothesis nodes*: standard deviation	*Number of independent diagnosis subproblems including current observations*
6	*Indegree current observation nodes*: standard deviation	*Indegree current observation nodes*: maximum
7	*Overlap current observation*: maximum	*Indegree current observation nodes*: standard deviation
8	*Overlap current observation with label*: average	*Overlap current observation*: standard deviation
9	*Overlap current observation*: average	*Shortest path*: standard deviation
10	*Indegree of effect nodes*: average	*Indegree of effect nodes*: maximum
11	*Outdegree of hypothesis nodes*: average	*Number of hypotheses*
12	*Indegree current observation nodes*: maximum	*Shortest path*: maximum
13	*Indegree of effect nodes*: standard deviation	*Indegree of effect nodes*: standard deviation
14	*Overlap current observation*: standard deviation	*Kolmogorov complexity*
15	*Covering*: average	*Number of causal relations*
16		*Number of effects*
17		*Overlap*: standard deviation
18		*Overlap with labels*: standard deviation
19		*Number of independent diagnosis subproblems*
20		*Overlap current observation*: average
21		*Indegree current observation nodes*: average

Table 6.8: Confusion matrix *Artificial Samples*.

		Predicted							
		ATMS	*CF*	*HS-DAG*	*HS-DAG$_{QX}$*	*XPLorer*	*XPLorer$_{QX}$*	*ASP*	**Total**
Actual	*ATMS*	**12**	0	16	8	0	0	0	36
	CF	0	**0**	0	0	0	0	0	0
	HS-DAG	0	0	**136**	33	0	0	0	169
	HS-DAG$_{QX}$	1	0	35	**83**	0	0	0	119
	XPLorer	0	0	3	3	**0**	0	0	6
	XPLorer$_{QX}$	0	0	1	1	0	**0**	0	2
	ASP	0	0	0	0	0	0	**0**	0
	Total	13	0	191	128	0	0	0	332

Table 6.9: Confusion matrix *FMEA Samples*.

		Predicted							
		ATMS	*CF*	*HS-DAG*	*HS-DAG$_{QX}$*	*XPLorer*	*XPLorer$_{QX}$*	*ASP*	**Total**
Actual	*ATMS*	**216**	1	10	17	0	0	0	244
	CF	1	**2**	0	0	1	0	0	4
	HS-DAG	24	0	**32**	14	0	0	0	70
	HS-DAG$_{QX}$	16	0	8	**84**	0	0	0	108
	XPLorer	0	0	0	0	**0**	0	0	0
	XPLorer$_{QX}$	0	0	0	0	0	**0**	0	0
	ASP	0	0	0	0	0	0	**0**	0
	Total	257	3	50	115	1	0	0	426

in *Artificial Samples*, we can see that the Precision value is close to the best value of 1, since it was only once selected incorrectly. However, the Recall value is rather disappointing due to various samples where *ATMS* was in fact the fastest method, but was not identified as such by the classifier. This trade-off is also apparent in the F_1-Score, which hence is only 0.49. In contrast in the *FMEA Samples*, Precision and Recall values are good and hence also the F_1-Score for *ATMS* is promising. Other common measures in multi-class classification are Accuracy, i.e., the overall effectiveness of a classifier, Specificity, i.e., how well negative labels are classified, and the area under the receiver operating characteristic curve (AUROC), i.e., the ability of the predictor to avoid false labeling [Wit+11]. From the contingency

Table 6.10: Classifier performance measures *Artificial Samples*.

	Precision	Recall	F_1-Score	Accuracy	Specificity	AUROC
ATMS	**0.92**	0.33	0.49	0.92	**1.00**	0.85
CF	0.00	0.00	0.00	**1.00**	**1.00**	-
HS-DAG	0.71	**0.80**	**0.76**	0.77	0.66	0.81
HS-DAG$_{QX}$	0.65	0.70	0.67	0.76	0.79	0.82
XPLorer	0.00	0.00	0.00	0.98	**1.00**	**0.98**
XPLorer$_{QX}$	0.00	0.00	0.00	0.99	**1.00**	0.55
ASP	0.00	0.00	0.00	**1.00**	**1.00**	-

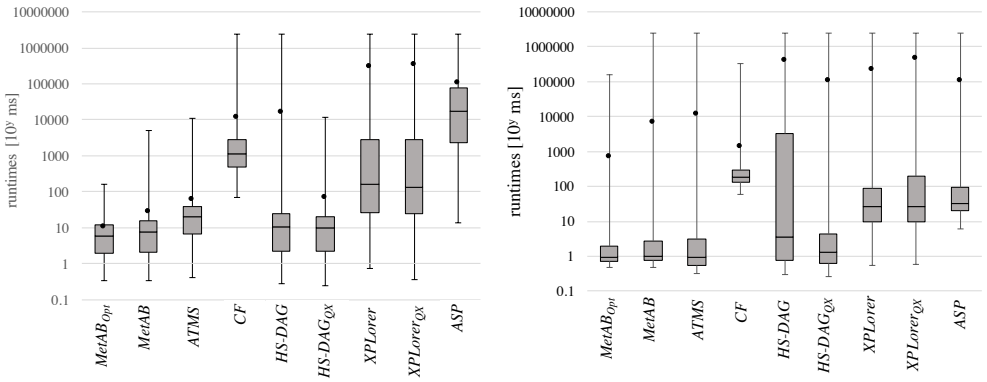
Table 6.11: Classifier performance measures *FMEA Samples*.

	Precision	Recall	F_1-Score	Accuracy	Specificity	AUROC
ATMS	**0.84**	**0.89**	**0.86**	0.83	0.77	0.91
CF	0.60	0.50	0.57	0.99	**1.00**	**0.99**
HS-DAG	0.64	0.46	0.53	0.87	0.95	0.88
HS-DAG$_{QX}$	0.73	0.78	0.75	0.87	0.90	0.92
XPLorer	0.00	0.00	0.00	**1.00**	**1.00**	-
XPLorer$_{QX}$	0.00	0.00	0.00	**1.00**	**1.00**	-
ASP	0.00	0.00	0.00	**1.00**	**1.00**	-

tables is apparent that *HS-DAG*, *HS-DAG$_{QX}$*, and *ATMS* are the best performing approaches with some variations between the sample sets. A premature analysis of these results would suggest that applying for instance *HS-DAG* to every artificial instance would yield the optimal runtime for most problems. However, based on the entire set of problems, i.e., test and training, *HS-DAG* is not the most efficient approach as its computation time is notably larger on several instances than other algorithms. Thus, we propose to use our meta-approach MetAB which we subsequently compare to always using a single diagnosis method.

Figure 6.9a and 6.9b depict the distributions of the log runtimes of the abductive reasoning algorithms on the test sets. In addition to the seven abduction methods, we have included on the one hand our meta approach *MetAB* and on the other hand an optimal algorithm selection approach *MetAB$_{Opt}$*. For *MetAB$_{Opt}$* we assume a classifier that labels each diagnosis problem correctly and thus computes the explanations with the approach requiring the minimal runtime. *MetAB$_{Opt}$*'s computation time thus consists of the collection of the online metrics, the classification, and the diagnosis using the fastest approach. For the artificial examples, *MetAB* and *MetAB$_{Opt}$* show the best average and median runtime results, with an average percentage difference between them of 15.30 %. However, In case of the FMEA examples, *ATMS* provides the best median results on the test set followed by *MetAB$_{Opt}$* and *MetAB*. This is unsurprising given the dominance of *ATMS* on *FMEA Samples*. On average, yet, *ATMS* is slower than *MetAB$_{Opt}$* and *MetAB*. The average percentage difference between *MetAB$_{Opt}$* and *MetAB* is 10.93 %.

Table 6.12 lists the runtime results of *MetAB* in comparison to *ATMS* and *HS-DAG$_{QX}$*. From the table we can observe that for the FMEA examples the classifier at least once chose an algorithm which cannot compute the diagnoses within the given time frame, hence, the maximum runtime corresponds to the penality of $\theta = 40$ *minutes* plus the classification time.



(a) *Artificial Samples* **(b)** *FMEA Samples*

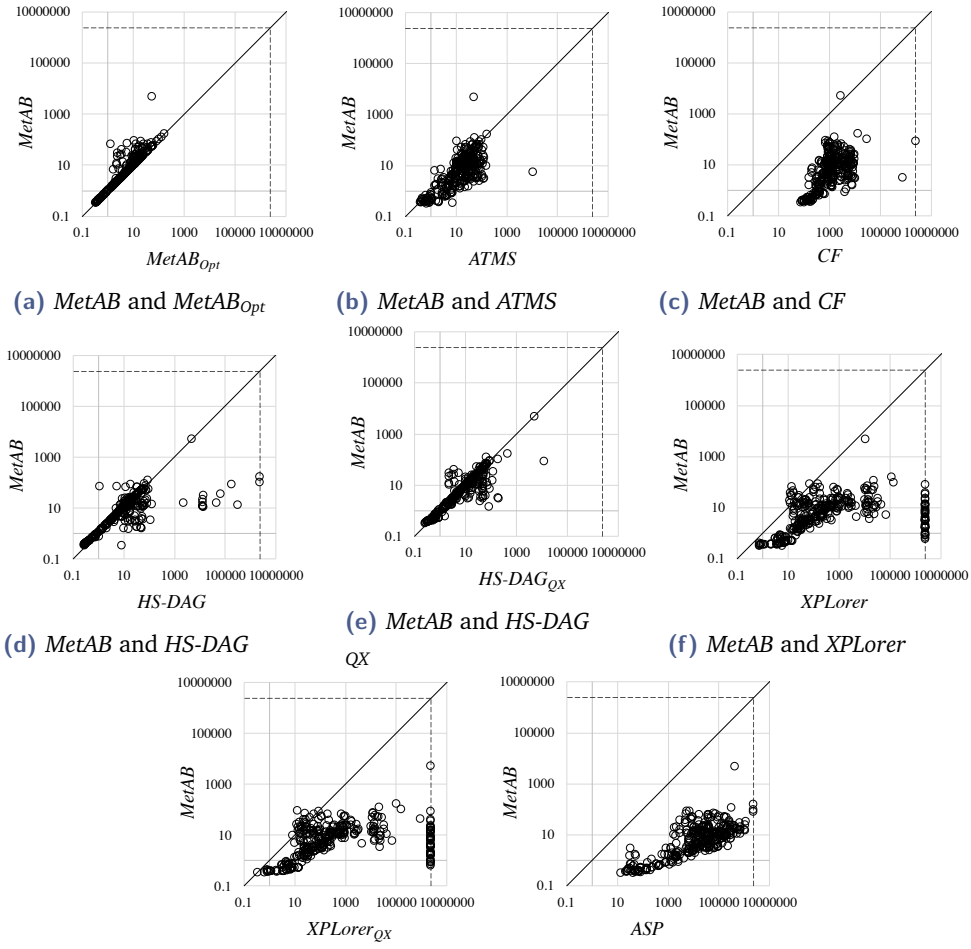
Figure 6.9: Statistical distribution for the runtimes [10^y ms] on the test set.

Table 6.12: Runtime results of the meta-approach in comparison to *ATMS* and *HS-DAG*$_{QX}$.

	Artificial Samples			FMEA Samples		
	MetAB	*ATMS*	*HS-DAG*$_{QX}$	*MetAB*	*ATMS*	*HS-DAG*$_{QX}$
MIN	0.32	0.34	**0.22**	0.48	0.33	**0.25**
MAX	**4,940.16**	170,838.66	2,400,000.00	2,400,000.23	**2,400,000.00**	**2,400,000.00**
AVG	**28.16**	226.28	10,269.27	**6,963.16**	14,239.04	124,552.12
MED	**7.38**	19.91	8.13	1.04	**0.91**	1.39
SD	**271.09**	4,529.92	148,023.42	**117,443.57**	174,655.60	531,157.15

For a more in-depth comparison of MetAB to the other diagnosis methods, we provide various runtime scatterplots in Figure 6.10 and 6.11. As apparent from Figure 6.10a and Figure 6.11a, *MetAB* performs not as good as the ideal approach *MetAB*$_{Opt}$. Nevertheless, given the classification accuracy most data points are on the diagonal or close to it confirming that the selected algorithm is either the fastest or among the best solvers for the *PHCAP*. For the FMEA examples, we can see that on one instance, *MetAB* choses an algorithm exceeding the time limit hence featuring a penalized runtime. From the remaining plots, we can conclude that while on some *PHCAP*s the meta-approach cannot compete with always choosing a single algorithm, the bulk of data points is usually located below the diagonal suggesting the benefit of *MetAB* even when an ideal prediction is not possible.

To evaluate the efficiency of our meta-approach in comparison to always choosing a single approach, we determine whether the median difference between *MetAB* and the best performing approaches on the corresponding sample set is significant. Since dealing with non-normal runtime distributions and under the assumption that the observations are independent and identically distributed, we apply the one-tailed Wilcoxon Signed-Rank Test [Wil45]. Suppose paired runtime data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ from *MetAB* and the compared abduction approach, respectively. We propose hypothesis $H_0 : m_X = m_Y$, stating a median difference of zero between pairs of observations. For *FMEA Samples*, we compared *MetAB* and *ATMS* and formulate our alternative hypothesis as $H_1 : m_X > m_Y$. Given the one-tailed test for $\alpha = 0.05$, we reject H_0, i.e., stating a significant difference between *MetAB* and *ATMS*. This indicates that on simple examples such as the one's generated based on FMEAs, choosing the *ATMS* as abduction procedure is advantageous on the test data. In regard to the artificial examples, we compared the meta-approach to *HS-DAG* and *HS-DAG*$_{QX}$ with $H_1 : m_X < m_Y$. In both cases, we accept the null hypothesis since the Wilcoxon signed-rank test determined an insignificant improvement of *MetAB* in comparison to the hitting set approaches.

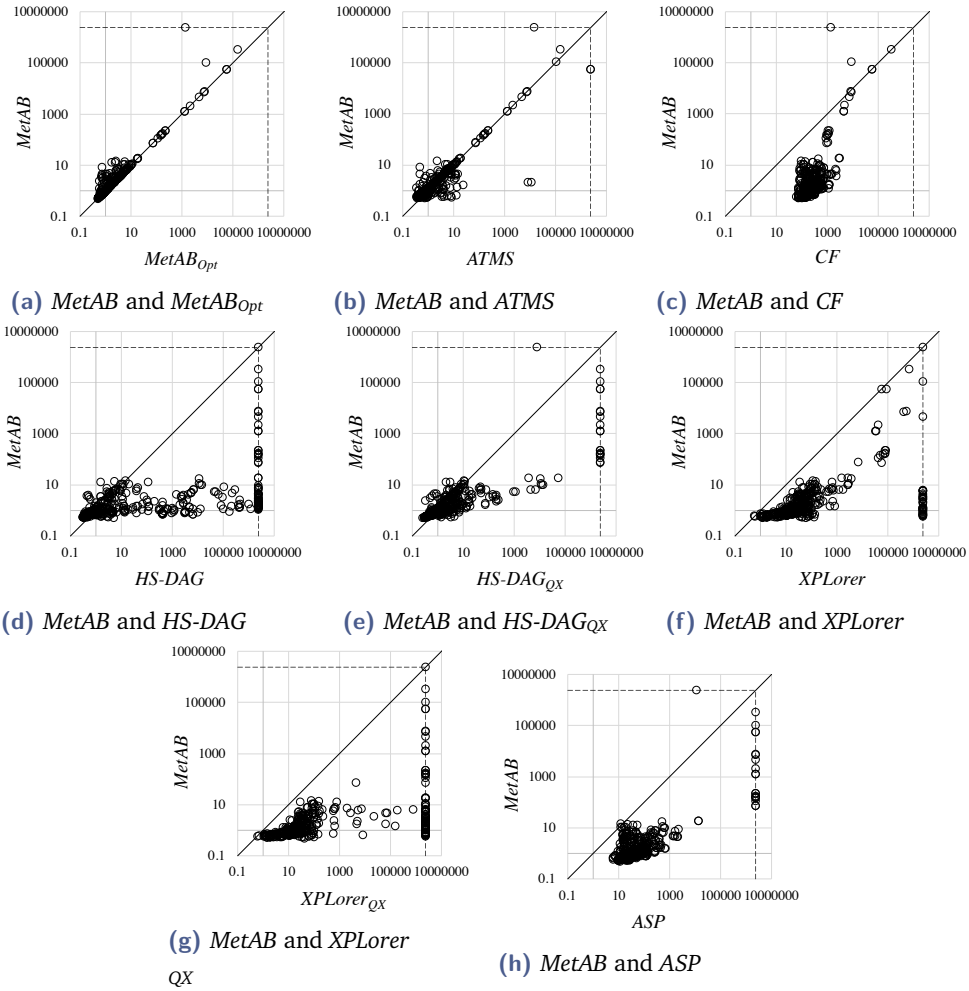


(a) *MetAB* and $MetAB_{Opt}$ **(b)** *MetAB* and *ATMS* **(c)** *MetAB* and *CF*

(d) *MetAB* and *HS-DAG* **(e)** *MetAB* and $HS\text{-}DAG_{QX}$ **(f)** *MetAB* and *XPLorer*

(g) *MetAB* and $XPLorer_{QX}$ **(h)** *MetAB* and *ASP*

Figure 6.10: Scatter plots of runtime [10^y ms] comparisons on the test set of *Artificial Samples*.



(a) *MetAB* and $MetAB_{Opt}$ **(b)** *MetAB* and *ATMS* **(c)** *MetAB* and *CF*

(d) *MetAB* and *HS-DAG* **(e)** *MetAB* and $HS\text{-}DAG_{QX}$ **(f)** *MetAB* and *XPLorer*

(g) *MetAB* and *XPLorer*
QX

(h) *MetAB* and *ASP*

Figure 6.11: Scatter plots of runtime [10^y ms] comparisons on the test set of *FMEA Samples*.

6.5 Conclusion

Algorithm selection has been proposed as a way to deal with the issue that given different problem instances there is no universally superior approach computing solutions efficient for all samples, but the performance of a certain method highly depends on the underlying characteristics of the problem instance. Abductive model-based diagnosis represents a novel application area of the portfolio approach. Utilizing a machine learning classifier and structural attributes of these system descriptions, we have developed a meta-approach to abductive model-based diagnosis that aims at predicting the most suitable method for a diagnosis problem on a case-by-case basis. Algorithm selection can be particularly useful in the context of model-based diagnosis, since the structural features of the underlying system description can be computed mostly offline and only the dynamic portion depending on the observations have to be derived during computation.

We tackle two abductive diagnosis frameworks with algorithm selection, namely **(1)** the simple bipartite abduction problems that are obtained from FMEAs and we have examined in Chapter 4 and **(2)** more expressive models, i.e., *PHCAP*s, from Chapter 5:

(1) We explore metrics inherent to the structure of FMEA-based models, which form the feature vector for a classifier as part of a meta approach. Subsequently, we compared abduction via the meta-approach to always choosing a single abductive reasoning mechanism. Evaluated on a test set, the metrics led to a satisfactory selection of the best algorithm for a particular diagnosis problem. In case of the FMEA-based samples our meta algorithm was restrained by the time necessary to construct the feature vector and thus could not outperform all abduction methods. Our approach shows its value when operating on larger problem instances, i.e., artificial examples, where it performs well and in fact is the most efficient on average in comparison. In the second evaluation, we exchanged the Boolean approach with Berge' s algorithm to allow an on the fly diagnosis computation. The empirical evaluation showed that the extracted properties of the instances allow to determine the "best" abduction method. Even in cases where the classification is incorrect, the approach selects the second most efficient algorithm and thus overall outperforms the other diagnosis methods. Therefore, we believe that this meta approach is a feasible alternative to continuously using a single abduction procedure.

(2) In the second portion of this chapter, we evaluated the algorithm selection method within an empirical set-up based on seven abductive reasoning methods for propositional Horn theories and two benchmarks. The accuracy of our algorithm selection technique is satisfactory and *MetAB* is in fact on average more efficient than choosing a single abductive reasoning approach on both sample sets. Hence, based on our benchmarks we can advocate for the benefit of using a portfolio method for abduction. While on average the fastest, our meta-approach cannot outperform all other abduction techniques on median on the simple diagnosis problems stemming from FMEAs. There the meta-approach's set-up effort might not justify the prediction. Possibly on larger models, we could observe a better performance of *MetAB* in comparison to *ATMS*, which is known to have difficultly propagating label values given larger more interconnected theories.

Thus, adapting and extending the benchmarks used for evaluation might provide improved runtime results for our portfolio approach. Even though we have applied a simple attribute selection, a deeper analysis of negligible or combinable features would definitely improve the technique since choosing the incorrect method can be an expensive mistake within our set-up.

Part III

Case Study: Wind Turbine Fault Identification

Wind Turbine Diagnosis

7

> *More solar energy falls on Earth in one hour than all the energy our civilization consumes in an entire year. If we could harness a tiny fraction of the available solar and wind power, we could supply all our energy needs forever, and without adding any carbon to the atmosphere.*
>
> **— Ann Druyan and Steven Soter**
> "*Cosmos: A Spacetime Odyssey*". 2014.

This chapter is based on the following publications:

- [Gra+15] Christopher S Gray et al. „An Abductive Diagnosis and Modeling Concept for Wind Power Plants“. In: *IFAC-PapersOnLine* 48.21 (2015), pp. 404–409
- [Koi+18] Roxane Koitz et al. „Wind Turbine Fault Localization: A Practical Application of Model-Based Diagnosis“. In: *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*. Cham: Springer International Publishing, 2018, pp. 17–43. ISBN: 978-3-319-74962-4. URL: https://doi.org/10.1007/978-3-319-74962-4_2

The majority of the chapter has been taken from [Koi+18], while Section 7.3 has been published in [Gra+15].

7.1 Motivation

The increasing complexity and magnitude of technical systems is leading to a demand for effective and efficient automatic diagnosis procedures to identify failure-inducing components in practice. This is especially true in application areas experiencing excessive service costs and idle time revenue loss. In the industrial wind turbine domain Operation and Maintenance (O&M) constitute significant factors in terms of turbine life expenditure. This industry has been expanding rapidly during the last fifteen years. Today many European nations produce a significant amount of wind energy [Ass14]. This expansion has been accompanied by an ongoing technological evolution, with the power rating, scale, and complexity of individual machines having increased continuously. However, the wind industry is under growing pressure to demonstrate profitability and offer energy at competitive costs. In the latter context, O&M represents a significant proportion of the life-cycle costs as turbines are located in isolated regions and weather conditions may further restrict access. Many cases are reported, where high failure rates of major systems such as the gearbox, yaw drive, or pitch drive resulted in extremely high repair costs. Given the isolated locations of wind turbine installations, accurate fault identification is essential for reducing costs and risks of component failures as well as turbine downtime. The use of remote detection and diagnostic technology is an area which is receiving an increasing level of focus, in particular in the offshore wind energy industry, where turbine failures are even more critical due to the difficulties related to accessing and repairing the machines in potentially harsh environmental conditions [GW10].

All modern wind turbines use sensors, data acquisition and on-board processing as part of the closed-loop control system. Furthermore, a range of diagnostic functions is typically

included within the system controller, so that at least basic status information can be provided in case of faulty operation. However, such on-board diagnostics are limited by the computing resources of the turbine controller and the absence of instant access to a long term historical database. Additionally, wind turbine diagnosis is complicated, since their overall reliability is affected by a multitude of failure modes concerning all major sub-systems and furthermore load conditions change regularly [Wil+10]. While electrical and control systems account for most wind turbine failures, other sub-systems, such as gearboxes, cause extensive downtimes due to the complexity of maintenance and thus pose a higher cost risk [Taz+17]. Unfortunately, currently implemented standard alarm systems deliver a large number of false alarms and thus are not suitable for standalone fault detection and identification [GW10].

Wind turbine operators often rely on time-based maintenance, where turbines are inspected periodically to assess their condition. This practice may lead to unnecessary turbine downtime for healthy systems, while failure-inducing conditions between services remain unnoticed. Due to these disadvantages predictive and condition-based maintenance have become increasingly popular. Both rely on software monitoring the turbines' operation and diagnosis methods [Lu+09]. Condition monitoring software utilizes the signals transmitted from the sensors integrated within the turbine and further processes the data to derive health information of critical components, e.g., gearboxes and main bearings. This is especially useful since, some specification indicators of a subsequent failure are observable prior to around 99% of equipment fault occurrences. Thus, unnecessary maintenance activities can be avoided by scheduling repair or replacement of components based on their present or impending failure risks and complex tasks requiring specialized human resources, tooling, or spare parts may be planned well in advance [Taz+17]. In order to support such diagnostic activities, available information such as Supervisory Control And Data Acquisition (SCADA) operational logs, service documentation, inspection results, and historical failure rates may be exploited. SCADA data are of special interest for wind turbine condition monitoring, as 10-minute values of different signals provide readily available information relating to turbine health [Tch+14].

Different fields such as control engineering have focused on methods to predict and identify failures through the usage of condition monitoring systems and advanced data analysis. However, an accurate and automated diagnosis of root causes for wind turbines is complicated because of the amount of component failure modes [Wil+10]. In addition, the effectiveness of such an approach depends strongly on an accurate definition of how subsystems may fail and the associated physical mechanisms. Therefore, many systems are only able to detect portions of failures from automatically retrieved data streams. Another common restriction in available condition monitoring applications, is to merely focus on specific error prone parts of the turbine, such as the gearbox. Hence, a system-level solution is required that is capable of building on the existing knowledge concerning wind turbine reliability. In addition, the approach should combine data and inspection derived state indicators, in order to provide robust diagnosis results.

The AMOR project between Uptime Engineering[1] and Graz University of Technology aims at integrating an abductive model-based diagnosis and modeling engine, created by the university, in consideration of the process shown in Chapter 3 into Uptime Engineering's wind turbine condition monitoring software. In this chapter, we first introduce state-of-the-art research on wind turbine fault identification and then present the type of information that is available and may be used as a basis in context of a diagnosis application. In particular, we introduce Uptime Engineering's comprehensive failure assessment of industrial wind turbines that provides a structured evaluation of faults and their manifestations that is

[1] Uptime Engineering provides consulting services as well as software tools in the field of technical reliability.

continuously extended and updated. This analysis can be exploited by the application as the basis of the model development phase to construct a suitable diagnostic system description. However, there are certain obstacles in the representation that hinder a straight forward compilation as we have seen based on FMEAs or FTAs. Hence, we describe this issue in detail and present a solution in Chapter 8. Chapter 9 then focuses on a model-based diagnosis application in development. In particular, we present a GUI design and workflow of a web application integrated into Uptime Engineering's software.

7.2 Related Work

Wind turbine reliability presents a very interesting use-case for the application of diagnostic methods. Condition monitoring and diagnosis are challenging tasks; however, they could substantially contribute to a minimization of system down-time. A large and growing body of literature has investigated fault prediction, detection, and identification in this domain that has led to numerous approaches. Signal processing techniques analyze the multidimensional turbine data without considering an a-priori developed mathematical model to extract faults based on spectral analysis or trend checking techniques, while machine learning methods, such as neural networks, can rely on historic data for failure identifications [Ham+09]. Zaher and McArthur [ZM07] present a preliminary multi-agent system for wind farms that comprises fault and degradation detection. By collecting data from various downtimes their application computes an overall turbine operational behavior model. Through the employment of supervised learning of the nominal behavior, the framework is able to identify power curve and temperature deviations. Their approach, however, is limited to a wind turbine's gearbox, generator, and rotor blades.

Many systems rely on SCADA data already available, as they provide a cost-effective source of condition monitoring information. Qiu et al. [Qiu+12] focus in their research on SCADA alarms and attempt to determine where alarms stem from and prioritize them. Based on their results, Chen et al. [Che+12] create a BBN to model the relationship between root causes and symptoms in wind turbines. While their use of BBNs allows a clear representation of cause-effect relationships, the complexity of the network increases exponentially with the amount of SCADA alarms considered. Schlechtingen, Santos, and Achiche [Sch+13] also adopt machine learning by creating neural networks based on the normal wind turbine behavior in combination with fuzzy rules representing expert knowledge on faults. Their approach requires the availability of SCADA operation logs, which provide 10-minute values of various measurements, such as power output, rotor speed, or gearbox oil temperature. Anomalies can be detected by comparing the normal-behavior model with the actual performance. The fuzzy inference system then automatically identifies the faulty components and further presents possible root causes for malfunctions. In contrast to many other approaches, their system is capable of detecting abnormal behavior in a large variety of components. Recently, Toubakh and Sayed-Mouchaweh [TSM16] have developed a data-mining approach to improve fault localization of hybrid dynamic systems, such as wind turbine converters. The approach defines a feature space for the continuous dynamics of the component that allows to define an indicator for converter degradation based on a drift in the normal behavior classes. The system is capable of detecting and identifying a present failure early on.

In the context of model-based reasoning, the research by Echavarria et al. [Ech+07] is worth mentioning. The authors developed a fault diagnosis, repair, and functional design system for offshore wind turbines. Their technique uses a model-based reasoner in conjunction with qualitative turbine models to identify faults. In order to increase wind turbine reliability, a functional redundancy designer is utilized, enabling the identification

of substitutes for different components. In case of a fault, the system shall provisionally reconfigure itself in order to compensate for the functionality loss, and allow the turbine to continue its work. Future research on the authors' side will incorporate the reconfiguration as a maintenance strategy. The overall goal is a self-maintaining wind turbine, which is independent from maintenance crews to some extent. Hameed et al. [Ham+09], Lu et al. [Lu+09], and Tchakoua et al. [Tch+14] provide systematic literature reviews on the topic of condition monitoring and fault diagnosis of wind power plants.

7.3 Model-Based Wind Turbine Diagnosis

An industrial wind turbine continuously stores operational data logs (SCADA logs), which can be retrieved and transferred to a central data store. The use of such data stores for detailed performance analysis and diagnostic work is becoming standard in the wind industry, since the data is readily available and provides information about a number of systems and components within the turbine. Today most medium to large scale operators of wind turbine fleets have installed centralized data management systems to collect and store such SCADA logs. Uptime Engineering has developed a software application that is capable of performing automated and continuous analysis of such data, typically with the aim of detecting anomalies in the behavior of individual turbines. Continuous advances have been made in the capabilities of the analytic models, and it is now possible to detect outlying behavior with a high degree of sensitivity. The results of such analysis are combined with the above-mentioned on-board diagnostic results, as well as general information concerning the turbine age, type and build status, in order to support the turbine operator in efficiently reacting to detected anomalies.

However, such analysis activities often produce a high volume of information (multiple turbines monitored, multiple alarms originating from many systems and accompanied by a range of heterogeneous supporting information). The main challenge facing the user of such a system is efficient interpretation of the results and the derivation of an effective response strategy. Therefore a strong need has been identified to provide the software user with "decision support"; i.e., an additional layer of intelligence built in to the software, which combines all generated observations and produces clear recommendations for action. Model-based diagnosis is a highly relevant solution, due to the strong capability of the approach in combining state information from a multitude of sources and identifying the highest likelihood root cause.

Uptime Engineering exploit a combination of diagnostic and prognostic techniques exploiting the relations between operational and environmental loads as also damage accumulation rates [GW10; Gra+11]. Based on an analysis of potential component-based failure modes and their damage driving physics, a mathematical model is computed that can be used to calculate the rate at which damage accumulates in response to the operating environment. This method offers a means for determining current and projected failure probabilities based on the derived damage model and statistical failure model. Statements about absolute remaining useful life can not be made, since the load capacity would have to be known in advance to a high degree of accuracy. This is very rarely the case, and considerable variation occurs due to, e.g., variations in material quality, tolerances in component manufacture, influence of transport, installation and configuration. Therefore the prognostic method focuses on quantification of the applied loads instead, and uses probabilistic methods to relate said loads to damage accumulation.

The basis of our model, which we discuss in detail in Section 8.4, is a structured Failure Mode Assessment (FMA) of different subsystems within a wind turbine provided by our

industrial partner. Experts continuously adapt and extend these records that contain each subsystem's key components and their possible physical and chemical failure modes. The assessment characterizes clear causal dependencies from specific faults to various measurements of the health monitoring system. Due to the fact that industrial wind turbines comprise a large variety of components and therefore an even greater amount of possible failures, the analysis focuses on malfunctions contributing most significantly to turbine down-time. These failure modes are represented as abnormality assumptions, hypotheses *Hyp*, within in an abduction system description, similarly as we have seen for the modeling of FMEAs.

Example 7.1

A converter is an essential component of modern industrial wind turbines, since it allows operation at variable speed whilst connecting to a constant frequency grid. Therefore faults of the converter result in costly turbine down-time. In addition, converter faults produce a large amount of SCADA alarms in the control center. Hence, being able to correctly diagnose converter faults and optimize maintenance should increase wind turbine availability to a great extent. Table 7.1 shows parts of the FMA produced by an expert group during a system analysis for the converter. Let us focus on the first table entry and its underlying principles: As the electrolyte capacitor of the buck boost system degenerates the equivalent series resistance rises (*ESR*), which causes the output voltage of the buck boost to drop. The lower voltage is fed into the inverter and consequently contributes to a reduced turbine power output (*P_turbine*). In addition, if the controller tries to adjust, the insulated-gate bipolar transistor (IGBT) becomes more active as it tries to compensate the output voltage, which causes a rise in the power cabinet temperature (*T_power_cabinet*) due to more switching of the IGBT. However, it is not ensured that both the turbine output would be decreased and the temperature in the power cabinet rises. Hence, these two manifestations are linked with a logical *OR*[a]. Hence the first table entry can be read as: Given the failure, electrical chemical aging of the converter and probably partial loads and a high ambient temperature, the temperature of the power cabinet would be higher than normal and/or the power of the turbine would be lower than expected, given the wind speeds. Further, when inspecting the buck boost's electrolyte capacitor, a higher equivalent series resistance can be measured[b].

Table 7.1: FMA of the converter.

Fault Mode	Component	Damage Promoting Operating Mode	Aggravating Boundary Conditions	State Indicators/Part Inspection	Maintenance Task
Electrical chemical aging	Buck Boost - Electrolyte Capacitor	Partial load	High ambient temperature	(T_power_cabinet *OR* P_turbine) *AND* ESR	Replace Capacitor
Corrosion	Fan - Pin	Full load	Saline environment, high temperature environment	T_cabinet *OR* P_turbine	Replace Fan
Thermo-mechanical fatigue (TMF)	Fan - Bearing Running Surface	Start Up/ Shut Down, Transient current/voltage events	Light winds (frequent start up and shot down), changing wind direction (yaw adjustment)	T_cabinet *AND* P_turbine *AND* Alarm-code_overvoltage_link	Replace Fan
High-cycle fatigue (HCF)	IGBT - Wire Bonding	Start Up/ Shut Down, Transient current/voltage events	Low ambient temperature	T_inverter_cabinet *OR* T_nacelle *OR* P_turbine	Replace all IGBTs

[a] We discuss in Chapter 8 how we handle this type of knowledge and how to automatically convert such a failure assessment to an abductive knowledge base.
[b] Measuring the equivalent series resistance is part inspection task, i.e., this data is not automatically retrieved from the turbine.

The observable anomalies refer to SCADA alarms, deviations in different SCADA data signals from normal values, and part inspection results. Alarms are directly sent to the control center without any filtering mechanism implemented and are activated in response to measured signals deviating from defined limits and hence indicating a change in the turbine status. A specific alarm may often be triggered by several root causes and therefore cannot be used as a means to directly indicate a single failure mode. Hence, the operator receives a vast amount of alarms, which are, however, not directly reducible to a certain fault mode. Possible SCADA alarms are

- *Turbine stopped*,
- *Lack of wind: Wind speed too low*,
- *Ice detection: Ice sensor*,
- *Over current inverter: L1 inverter 1*,
- *Pitch control error: Limit switch 95° blade A*, or
- *Fault air cooling: Pressure too high*.

SCADA data signals are obtained as time series in contrast to discrete alarms and therefore have to be preprocessed accordingly. Once discrepancies are detected, they are translated into qualitative values suitable for our diagnosis engine, e.g. *P_turbine* stating that the power output is low in comparison to what it should be[2]. The signals originate from different sensors placed throughout the turbine, measuring, e.g., temperatures or currents of various components, power output, or wind speed and direction. Table 7.2 shows some exemplary SCADA data.

Table 7.2: Exemplary SCADA data types.

Variable	Unit
Power output	kW
Wind speed	m/s
Wind direction	°
Ambient temperature	° C
Rotor speed	rpm
Gearbox oil temperature	° C

Some fault modes can be confirmed by an inspection task. This additional information gained through visual checks or specific measurements at the wind turbine can be used to discriminate among the possible root causes. Furthermore, the assessment considers conditions that increase fault likelihood such as damage promoting operation modes, and aggravating environmental conditions. A Physics of Failure (PoF) approach [GW10] quantifies the relation between operation modes and damage accumulation. In addition, load scenarios in conjunction with historical data can be used to compute prognostic information on the remaining life expectancy of certain components. These conditions are excluded from the system description we utilize for abductive reasoning, but should be utilized as inputs to an additional ranking measure in upcoming versions of the diagnostic application. We

[2] Throughout the AMOR project, we have spent a considerable time in defining a taxonomy that is suitable for mapping the turbine data to qualitative values that not only are appropriate for the wind turbine domain, but may also be applied to other domains such as vehicle fleets.

give more insights into the PoF approach and the conversion of an FMA to an abductive knowledge base in Section 8.4 .

7.4 Conclusion

Industrial wind turbines are becoming more complex with diverse failure modes affecting components throughout the system. The cost of electrical energy produced depends strongly on the operational efficiency of the machines as well as the availability. Component faults leading to unplanned downtime have been shown to impact the overall energy production significantly, therefore the financial motivation for optimization in this respect is high. Yet, many of the proposed diagnosis systems merely focus on parts of the turbine or only locate a portion of the faults.

In order to create an appropriate model capable of revealing root causes for general state indicators, a deep knowledge of the internal workings and component interactions is necessary. Hence, our modeling approach relies on the availability of domain expertise on the behavioral changes in case of a fault. Since such an extensive expert assessment of potential component failures exists at Uptime Engineering, model-based diagnosis presents itself as a fault identification approach. Hence, we propose a method which takes advantage of available knowledge.

Given the model-based diagnosis, there is no need for additional sensor installations or complex data analysis and extraction from the fault identification side. Thus, the fault localization only requires some relatively minor efforts in extracting the models from the FMA, which we show in detail in the next chapter. The well known computational drawbacks of the abduction do not impair our troubleshooting activities, as the turbine health information stems from SCADA data and the frequency of diagnosis is relatively low, e.g., once per day. Furthermore, the amount of components to be considered for fault identification is manageable, since the maintenance task mostly does not consider the smallest replaceable unit, but higher-level components.

8 Extending the Modeling Framework for Abductive Diagnosis beyond Horn Clauses

> *It has long been an axiom of mine that the little things are infinitely the most important.*
>
> — **Sir Arthur Conan Doyle**
> "*A Case of Identity*". 1891.

This chapter is based on the following publications:

- [KW16b] Roxane Koitz and Franz Wotawa. „Integration of Failure Assessments into The Diagnostic Process“. In: *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society*. 2016, pp. 117–128

The extended modeling technique and its evaluation have not been published yet. The Physics of Failure approach and the generation of a suitable model thereof has been published in [KW16b].

8.1 Motivation

We have introduced in the previous part, a general process for abductive model-based diagnosis that makes use of information available after safety assessments of systems, e.g., FMEA and FTA. There the FMEA table or fault tree is automatically compiled into an abductive diagnosis model. In order to create a system description suitable for fault identification, two characteristics of an appropriate modeling language are vital: (1) it is capable of expressing information necessary for troubleshooting and (2) allows for efficient diagnostic reasoning. The latter is of particular importance as computing subset minimal abductive explanations for arbitrary models or theories in clausal form is located at the second level of the polynomial hierarchy, while for Horn theories the complexity is lowered by one level [EG95]. Although Horn clauses seem to be a reasonable limitation on the system description providing computational benefits [CT91], for practical applications this restriction may be too rigorous as for some domains it is necessary to express disjunctive effects within the diagnostic models.

Example 8.1

The following example motivates the need for a more elaborate logic modeling language beyond Horn clauses. Assume that we want to start our car for going on vacation. When turning on the ignition we can hear the starter working but the car's engine is not running. We know that we need the ignition switch to be at state one, the battery must not be empty, there needs to be enough fuel in the tank, and the car's engine should not be faulty (including the attached cables etc.). Using this information, we conclude that there might be a lack of fuel or the engine is broken. However, we would rule out an empty battery

because the starter is working requiring the battery to be charged. As experience has shown the last conclusion is not right. There might be enough power in the battery for the starter but not enough for the spark plug necessary to run the engine. Hence, in this case, the state empty battery causes two effects; either the starter is not working or there is not enough electricity to cause the spark plugs to fire. Given our observations that the engine is not running but we can hear the starter working, we conclude that the battery has to be empty. Yet given the traditional framework of abduction, an empty battery would not be considered an explanation for our symptoms. This does not coincide with human reasoning, where given two alternative effects for a fault, in case one manifestation is present we would consider the fault a possible explanation.

Other approaches have dealt with more expressive representations such as Disjunctive Logic Programming (DLP). DLP has been proposed as an extension of logic programming, allowing disjunctions within the head of a rule. Eiter, Faber, Leone, and Pfeifer [Eit+99] introduce the `dlv` system, supporting abductive diagnosis based on DLP under stable model semantics. Further, the authors provide a transformation of different diagnosis problems into a disjunctive logic program suitable for the abductive reasoning task. Sakama and Inoue [SI00] present a reduction of abductive programs to disjunctive programs, while You, Yuan, and Goebel [You+00] describe how abductive reasoning can be used for DLP. Based on these results, diagnosis problems formulated as abductive programs can be solved using DLP systems [DK02]. Besnard, Cordier, and Moinard [Bes+08] extend abductive reasoning by defining an ontology-based inference system that can be used for deriving causal explanations. Based on formal inference patterns, abstraction relations defined by ontological information, and causal statements, their approach determines tentative explanations and has been implemented successfully using the dlv system [Moi10]. Yet these approaches do not result in the diagnoses we are seeking to obtain, considering our example from before.

The request for enabling a more expressive logic than Horn sentences stems directly from industry. We are currently working together with our industrial partner, who provides consulting in the field of technical reliability and develops condition monitoring software for the industrial wind turbine domain. According to their failure assessments, it can occur that component faults affect a number of system variables. These symptoms do not necessarily co-occur, but represent possible effect alternatives. That is, a fault may cause all of its consequences at once or just a subset of them. This distinction is essential from a practical point of view as it avoids discriminating root cause candidates rashly given not every symptom is present. A similar requirement can arise in case failure effects are temporally delayed from one another. Imagine the diagnosis computation is triggered while a failure has not entirely manifested itself within the system, i.e., not all effects are observable at the time the fault identification process is invoked. Then given the detected anomalies and a Horn representation of the system behavior, the failure would not constitute a diagnosis. Furthermore, noise and uncertainty in the measurement can lead to unreliable observations, i.e., a failure may have occurred but only a subset of its effects can be detected. Note here that the temporal aspect or uncertainty are not our main concern, but we only list these as possible scenarios benefiting from allowing effect alternatives[1]. An implementation of the subsequently presented methods has been integrated into our industrial partner's wind turbine condition monitoring software and is currently being evaluated during operation in regard to its feasibility as a fault localization approach.

[1] Further, directly considering these dimensions within the model would significantly increase the complexity of the fault identification task, which we aim to avoid. Yet, for the interested reader there is a large body of research on time and uncertainty in regard to diagnosis, e.g., [Bru+98].

In this chapter, we focus on logic modeling beyond Horn clause representations that are good enough for practical applications but avoid to increase running time substantially. By allowing in particular disjunctions within the head of the clauses, more expressive theories can be constructed. From our industrial partner's and our experience this relaxation of the logical restriction of the theory allows for a more accurate description of fault-effect relations in real world applications. With the aid of a mapping function, these extended representations are converted to Horn models that can be used with any abductive reasoner capable of handling Horn clauses. Given such a constructed Horn theory, a set of observations, and a reverse mapping function, we show that the diagnoses generated on the converted model—though not semantically sound—provide intuitive solutions similar to human-like reasoning. We argue that this is an essential benefit of our approach, since especially in practice, the acceptance of AI systems benefits from comprehensible problem solving. Our goal in this chapter is to reveal whether the converted Horn models, which are increased in size, avoid the additional computational effort associated with more expressive representations and still allow deriving diagnoses in a practicable time frame. In particular, we aim at identifying if diagnosis on the compiled model using a Horn clause theorem prover tailored specifically for this task is more efficient or whether a state-of-the-art general-purpose solver can achieve a more convincing performance. Hence Section 8.3 provides an initial empirical evaluation. Lastly in Section 8.4, we give insights into our industrial partner's failure assessment method that takes advantage of *PoF*. In contrast to FMEAs, a *PoF* analysis considers component failures in the context of operational/environmental loads and damage accumulation rates. Their assessment enables the description of a failure, such that logical conjunctions and disjunctions of manifestations can be expressed. Hence, we show analogous to Section 3.3 how we can automatically compile their documents into a system model.

8.2 Extended Modeling

In the following, we discuss how abductive theories using more elaborate propositional languages can be mapped to *PHCAP*s. Particularly for practical applications an extended expressiveness is useful and often required to represent the underlying system. In contrast to approaches extending the fundamental reasoning mechanism [DK86a], we focus on determining a "suitable" mapping of such an extended modeling language $\mathfrak{L}$ to Horn clauses. By doing so, we can rely on existing abductive diagnosis reasoning systems, whose practicability has already been shown [KW15c]. Our suggested conversion is appropriate in the sense that it allows to express disjunctions of effects in the underlying model and even though the transformation is not entailment preserving the explanations obtained from the compiled model are more rational from a human cognition view.

Example 8.1 (cont.)

Considering the example from the introduction we know that the empty battery in the car has two potential consequences. Either the starter is not working (and thus producing no noise) or the available electrical power is not enough to fire the spark plugs. Ignoring the underlying reasoning chain, we can distinguish those two cases both leading to the effect that the car's engine is not running. The following sentence represents the corresponding cause-effect relationship (R):

$$emptyBat \rightarrow \begin{pmatrix} (noStarterNoise \wedge engineNotRun) \\ \vee (starterNoise \wedge engineNotRun) \end{pmatrix}$$

In our scenario from before we can observe that the starter is working, but the engine is not running, i.e., $Obs = \{starterNoise, engineNotRun\}$. Given the observations as well as the described cause-effect relation R we would conclude that the battery is empty, i.e., $\{emptyBat\}$ is a diagnosis. However, this is a fallacy. Examining the entailment relation we notice that

$$emptyBat \cup R \not\models \{starterNoise \wedge engineNotRun\},$$

that is, the empty battery is not an explanation for our observed symptoms due to the disjunctions in the head of the clause.

From this example we conclude that the entailment operator is more restrictive in admitting explanations than human abduction given a non-Horn background theory. Relaxations of entailment have been proposed in the context of ASP under stable model semantics, where a *brave* entailment operator is introduced, such that observations have to be witnessed by at least one stable model in contrast to all stable models in traditional (*cautious*) entailment [Eit+99]. Yet in our framework, such a definition is too general and leads to incorrect explanations. In practice, though, ideally abductive reasoning systems come to similar conclusions as domain experts would. Therefore, our conversion from more expressive representations to Horn actually allow us to infer such intuitive diagnoses (e.g., *emptyBat* is an explanation given *starterNoise* and *engineNotRun*), which do not conform to classical abductive inference but provide more realistic solutions.

The trade-off between the expressiveness of the representation language and its tractability has been a recurring topic within AI research. General propositional reasoning is associated with extensive computational effort, while limited knowledge descriptions are often too restrictive for practice [DP91]. Therefore, Selman and Kautz [SK96] propose knowledge compilation. A given representation is compiled into a restricted target language offline, which allows for answering queries online efficiently. Since an exact translation is not always feasible, the authors use an approximation to best convey the information stored in the original theory. In particular, they introduce Horn approximation, where Horn clauses function as lower and upper bounds for the set of models of the original theory. Based on different compiled representations investigated, Darwiche and Marquis [DM02] introduce a knowledge compilation map, describing different target languages as well as the queries that can be performed in polynomial time. An approach applicable to diagnosis creates an incremental approximation which decreases the size of the compiled representation by covering only preferred solutions, e.g., most probable diagnoses [VP08]. Console, Portinale, and Dupré [Con+96] utilize compilation as a method to focus the abductive diagnostic process by avoiding the generation of diagnosis aspirants that are removed later on and by generating solution candidates that allow the discrimination of other contenders.

Within our mapping, we restrict the source representation to the modeling language $\mathfrak{L}$, which ensures that we are able to compile a theory written in $\mathfrak{L}$ into a Horn clause representation. Hence, there is no need for approximating the target encoding. The principal idea behind our approach is best described in Figure 8.1. There we have a theory $Th_{\mathfrak{L}}$ written in a subset $\mathfrak{L}$ of propositional logic. We use a function η to map sentences written in $\mathfrak{L}$ to a Horn clause theory Th. Computing abductive diagnoses from Th leads to the set of all diagnoses Δ^S. When mapping this set back to the original hypotheses space using a function $\hat{\eta}$, we obtain solutions ($\Delta^S_{\mathfrak{L}}$) which are not equivalent to the diagnosis results from $Th_{\mathfrak{L}}$, but are closer to human expectations of diagnoses.

In Figure 8.1 we only consider η to be applied to a theory. In the following we extend this and allow to add new hypotheses to Th in order to represent $Th_{\mathfrak{L}}$ in a correct way. For

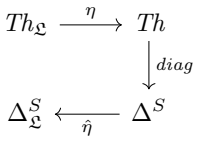


Figure 8.1: Mapping from one theory to another creating intuitive diagnoses.

the propositional modeling language $\mathfrak{L}$, we assume to capture Horn clause logic with the following two extensions:

Conjunctions at the right side of a rule. For the extended logical representation we allow $\mathfrak{L}$ to comprise rules of the form $p_1 \wedge \ldots \wedge p_n \rightarrow q_1 \wedge \ldots \wedge q_m$ where all p_i and q_j are elements of A.

Disjunctions at the right side of a rule. We further allow theories written in $\mathfrak{L}$ to include rules of the form

$$p_1 \wedge \ldots \wedge p_n \rightarrow \left(\begin{array}{c} (q_{1,1} \wedge \ldots \wedge q_{m_1,1}) \vee \ldots \\ \ldots \vee (q_{1,k} \wedge \ldots \wedge q_{m_k,k}) \end{array} \right) ,$$

where all propositions are elements of A and there exists exactly one p_i in $i = 1, \ldots, n$ that is also element of *Hyp*. Further we require that all $q_{1,1}$ to $q_{m_k,k}$ are to be in $A \setminus Hyp$.

The first extension can be simply handled via introducing a rule for each proposition used at the right side of an implication. The second extension requires more care when translated to a Horn theory because of the involved disjunction. The idea behind the representation can be best explained using the example from before, where

$$emptyBat \rightarrow \left(\begin{array}{c} (noStarterNoise \wedge engineNotRun) \\ \vee (starterNoise \wedge engineNotRun) \end{array} \right) .$$

When considering that those two classes of effects are due to a slightly different state of the battery, i.e., the battery is not charged versus the battery is not charged enough, we are able to represent the above sentence using two different assumptions as follows:

$$emptyBat_1 \rightarrow (noStarterNoise \wedge engineNotRun)$$
$$emptyBat_2 \rightarrow (starterNoise \wedge engineNotRun)$$

In this representation we explicitly refer to the two slightly different causes. By doing so we refrain from prematurely discriminating the original hypothesis in case one of the conjunctions validates to false.

To map a model in $\mathfrak{L}$ to Horn clauses, we have to convert $KB(A,Hyp,Th_{\mathfrak{L}})$ to $KB(A',Hyp',Th)$ by transforming the theory with the two mentioned extensions into Horn and adapting the set of hypotheses as well as propositional variables accordingly. Assume a $Th_{\mathfrak{L}}$ written in $\mathfrak{L}$. $Th_{\mathfrak{L}}$ contains a set of Horn clauses (*HC*) as well as a set of rules with conjunctions on the right hand side (*Conj*), and rules with disjunctions at the right hand side (*Disj*). We define the mapping function $\eta(Th_{\mathfrak{L}})$ resulting in the Horn theory *Th* as follows:

$$\eta(Th_{\mathfrak{L}}) = HC \cup \bigcup_{c \in Conj} \eta(c) \cup \bigcup_{d \in Disj} \eta(d) \tag{8.1}$$

The Horn clause sentences do not require any adaptation. As stated earlier, the implications stemming from the first extension can be translated into Horn clauses in a straightforward

manner; for each proposition q_j in the head of the rule a new clause is created where q_j is the only element on the right side. Thus, $\forall c \in \mathit{Conj}$:

$$\eta(c) = \bigcup_{j=1}^{m} p_1 \wedge \ldots \wedge p_n \rightarrow q_j \tag{8.2}$$

Handling the second extension requires us to convert the rules of the theory and adapt the set of hypotheses and variables. We define the second extension as clauses with a disjunction on the right hand side not containing any hypotheses and we require the body of the rule to comprise exactly one hypothesis and zero or more other propositions. Assuming $p_1 \in \mathit{Hyp}$, we describe the conversion for all $d \in \mathit{Disj}$ as

$$\eta(d) = \bigcup_{l=1}^{k} p_{1,l} \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q_{1,l} \wedge \ldots \wedge q_{m_l,l} \ . \tag{8.3}$$

For each conjunction l on the right side of d a new rule is added such that the conjunction constitutes the head of the rule. Moreover, the hypothesis in the body is exchanged by a new assumption, i.e., $p_{1,l}$, while all other propositions on the left side remain unchanged. This newly created hypothesis has to be considered in Hyp' and A'.

Depending on the number of elements within the conjunction, applying this conversion may lead to rules in the form of the first extension. Hence, we adapt $\eta(d)$ to

$$\eta(d) = \bigcup_{l=1}^{k} \eta(c_l) \ , \tag{8.4}$$

where c_l is $p_{1,l} \wedge p_1 \wedge \ldots \wedge p_n \rightarrow q_{1,l} \wedge \ldots \wedge q_{m_l,l}$.

Example 8.2

Let us consider a simple example to show the conversion using $\eta(\mathit{Th}_{\mathfrak{L}})$. Assume a knowledge base $\mathit{KB}(A,\mathit{Hyp},\mathit{Th}_{\mathfrak{L}})$ with

$$\mathit{Hyp} = \left\{ \ H_1, H_2, H_3 \ \right\},$$

$$A \ = \left\{ \ H_1, H_2, H_3, e_1, e_2, e_3, e_4, e_5 \ \right\},$$

$$\mathit{Th}_{\mathfrak{L}} = \left\{ \begin{array}{c} H_1 \wedge e_1 \rightarrow e_2, H_1 \wedge H_2 \rightarrow e_3 \wedge e_5, \\ e_2 \rightarrow e_3, H_3 \rightarrow e_1, H_3 \rightarrow e_4 \vee (e_3 \wedge e_5) \end{array} \right\}.$$

The second clause of the theory is an example of the first extension, i.e., $\mathit{Conj} = \{H_1 \wedge H_2 \rightarrow e_3 \wedge e_5\}$, the last clause features a disjunction on the right side, i.e., $\mathit{Disj} = \{H_3 \rightarrow e_4 \vee (e_3 \wedge e_2)\}$, and the remaining clauses are Horn, i.e., $\mathit{HC} = \{H_1 \wedge e_1 \rightarrow e_2, e_2 \rightarrow e_3, H_3 \rightarrow e_1\}$. Applying $\eta(\mathit{Th}_{\mathfrak{L}})$ results in the following $\mathit{KB}(A', \mathit{Hyp}', \mathit{Th})$:

$$\mathit{Hyp}' = \left\{ \ H_1, H_2, H_3, H_{3,1}, H_{3,2} \ \right\},$$

$$A' = \left\{ \ H_1, H_2, H_3, H_{3,1}, H_{3,2}, e_1, e_2, e_3, e_4, e_5 \ \right\},$$

$$\mathit{Th} = \left\{ \begin{array}{c} H_1 \wedge e_1 \rightarrow e_2, H_1 \wedge H_2 \rightarrow e_3, \\ H_1 \wedge H_2 \rightarrow e_5, e_2 \rightarrow e_3, H_3 \rightarrow e_1, \\ H_{3,1} \rightarrow e_4, H_{3,2} \rightarrow e_3, H_{3,2} \rightarrow e_5 \end{array} \right\}$$

Once the *KB*(*A*,*Hyp*,$Th_{\mathfrak{L}}$) has been converted to *KB*(A', Hyp', Th) and given a set of observations the diagnoses of *PHCAP*(A', Hyp', Th, Obs) can be computed as the set Δ^S using any diagnosis engine capable of processing Horn clauses. It is apparent that the hypotheses in Δ^S are not necessarily part of the original *Hyp* due to the compilation based on the second extension. In order to retrieve the desired diagnosis set $\Delta^S_{\mathfrak{L}}$, **(1)** Δ^S has to be mapped back into the original hypotheses space using $\hat{\eta}$ and **(2)** possible inconsistencies have to be resolved:

(1) To determine the original assumptions we have to map each newly generated hypothesis $\hat{H}$, i.e., $\hat{H} \in Hyp' \wedge \hat{H} \notin Hyp$, involved in a diagnosis back to its original hypothesis contained in *Hyp*. We define the function $\hat{\eta}$ as

$$\hat{\eta}(p_{i,l}) = p_i \ , \tag{8.5}$$

where $p_{i,l} \in Hyp' \wedge p_{i,l} \notin Hyp$. That is, each assumption added during the translation is mapped back to its original source hypothesis, while no transformation is necessary in case the proposition is already present in *Hyp*.

To retrieve the intuitive diagnoses of the original theory, each hypothesis in each diagnosis in Δ^S has to be converted in case it is not in *Hyp*. Since there might be several corresponding $\hat{H} \in Hyp'$ for each $H \in Hyp$, the mapping to the originals may lead to non-minimal solutions, i.e., supersets of other diagnoses. Therefore, we introduce a minimization function μ that removes all supersets to ensure parsimonious diagnoses:

$$\hat{\eta}(\Delta^S) = \mu(\{ \bigcup_{\hat{H} \in \Delta_x} \hat{\eta}(\hat{H}) \mid \Delta_x \in \Delta^S \}). \tag{8.6}$$

Example 8.2 (cont.)

Given the *PHCAP*(A', Hyp', Th, Obs) with $Obs = \{e_3, e_4\}$ we compute the diagnoses and obtain $\Delta^S = \{\{H_{3,1}, H_{3,2}\}, \{H_1, H_3, H_{3,1}\}, \{H_1, H_2, H_{3,1}\}\}$. Right away we can observe that newly created hypotheses are part of the diagnoses, consequently, these have to be converted back to their source hypotheses using $\hat{\eta}$. Considering the first solution we see that it contains two hypotheses which will be transformed back to the same proposition. Hence, diagnoses may decrease in size when translated back; in this case the double fault on the Horn model becomes a single fault diagnosis when mapped back. Converting the diagnoses back without minimization would result in $\{\{H_3\}, \{H_1, H_3\}, \{H_1, H_2, H_3\}\}$. Hence to retrieve parsimonious solutions an additional step is required where all supersets are removed by μ leading to $\hat{\eta}(\Delta^S) = \Delta^S_{\mathfrak{L}} = \{\{H_3\}\}$.

(2) It is possible that inconsistent explanations are derived due to the introduction of new hypotheses when applying η. Yet, only consistent solutions are valid, thus, any contradiction inducing diagnoses have to be disregarded.

Example 8.3

Given a *KB*(*A*,*Hyp*,$Th_{\mathfrak{L}}$) with

$$Hyp = \{ \ H_1, H_2 \ \},$$
$$A \ = \{ \ H_1, H_2, e_1, e_2, e_3 \ \},$$
$$Th_{\mathfrak{L}} = \{ \ H_1 \wedge H_2 \rightarrow \bot, H_1 \rightarrow e_1, H_2 \rightarrow e_2 \vee e_3 \ \}$$

and a set of observation $Obs = \{e_1, e_3\}$, applying $\eta(Th_{\mathfrak{L}})$ to the theory as described above we retrieve

$$Hyp' = \left\{ \; H_1, H_2, H_{2,1}, H_{2,2} \; \right\},$$

$$A' = \left\{ \; H_1, H_2, H_{2,1}, H_{2,2}, e_1, e_2, e_3 \; \right\},$$

$$Th = \left\{ \begin{array}{c} H_1 \wedge H_2 \rightarrow \bot, H_1 \rightarrow e_1, \\ H_{2,1} \rightarrow e_2, H_{2,2} \rightarrow e_3 \end{array} \right\}.$$

The abductive explanation given $PHCAP(A', Hyp', Th, Obs)$ is $\{H_1, H_{2,2}\}$. Transforming this solution back to the original hypotheses space results in $\hat{\eta}(\Delta^S) = \{\{H_1, H_2\}\}$, which is inconsistent with the theory and thus does not constitute a diagnosis. Hence, solution candidates inconsistent with the theory, i.e., $\forall \Delta_x \in \hat{\eta}(\Delta^S) : \Delta_x \cup Th \models \bot$, have to be removed subsequently[a].

[a] In case an ATMS is utilized for computing the explanations, it already provides a record of propositions that cannot be true simultaneously, i.e., the `NOGOOD` node.

The mapping function η obviously increases the size of the *PHCAP* both in terms of the number of literals and hypotheses. The increase is polynomial as stated in the following theorem.

Theorem 8.1. Let N_c and N_d be the number of rules comprising conjunctions and disjunctions respectively. Further let M_c and M_d be the maximum number of conjunctions and disjunctions occurring in any rule. The number of new rules is in the worst case proportional to $O(N_c \cdot M_c + N_d \cdot M_c \cdot M_d)$. The number of new hypotheses is in the worst case of order $O(N_d \cdot M_d)$.

Proof. In case of conjunctions on the right side of a rule, we add M_c new rules to the system. Hence, we have $N_c \cdot M_c$ new rules here in total. In case of disjunctions, we add M_d new rules comprising at the maximum M_c propositions at the right side. This leads to $N_d \cdot M_c \cdot M_d$ new rules in total. Adding these two worst case figures leads to the total number of added rules. Because of the conversion of disjunctions appearing on the right side of a rule, we add at the maximum M_d hypotheses for each such rule, leading to $N_d \cdot M_d$ new hypotheses in total.

8.3 Empirical Evaluation

There are certain practical application areas, where the restriction to a Horn model might not be able to grasp the entirety of how failures affect the system. We have given a simple example in the introduction, where a fault leads to two possible scenarios connected by a disjunction. The need for a more expressive system description has also been stated by industry, where failure assessments often not only record effects appearing in conjunction, but also comprise knowledge about symptom alternatives. This can be further useful in cases where there is a time difference between the occurrence of various manifestations or the anomaly detection method is subject to measurement inaccuracies. Furthermore, classical abductive inference does not conform with real world diagnostic reasoning in case disjunctive effects are involved as shown previously. Yet, it remains to show whether the increased theory size due to the transformation to Horn sentences induces runtime disadvantages or the computation effort remains practical. Subsequently, we provide first experimental results.

In the previous section, we have seen that the conversion from a model in $\mathfrak{L}$ to Horn clauses increases the size of the *PHCAP*. To determine whether this has consequences for practical applications, we carried out an empirical evaluation based on artificially generated examples representing theories written in $\mathfrak{L}$. Before describing the abduction methods and the needed modifications for general-purpose solvers, we report on how the $\mathfrak{L}$ models were created.

Our example generator constructs theories given a set of parameters: n defines the maximum number of hypotheses, o determines the upper limit of overlap between rules (i.e., the maximal number of subsequent rules sharing effects), r controls the maximum number of elements in the body, s describes the threshold for the number of literals in a conjunction within the head of a rule, and t is the maximum of propositions within the head of rules of the second extension. We randomly assign these propositions into conjunctions which are then connected by disjunctions. All these parameters are upper limits, e.g., the generator produces rules which have between 1 and r propositions in the body etc. The final number of rules and effects vary depending on the parameters. To construct a *PHCAP* the generator randomly chooses between 1 to k observations from the effect set. Furthermore, whenever an effect is contained within the body of a rule it is added as a fact to the theory. For our evaluation we created 144 *PHCAP*s, where every sample was generated with $r = s = o = 5$ and $t = 6$, while the number of observations k was randomized between 1 and 30. For simplicity, we chose only positive observations, to ensure that the computed diagnoses represent what we understand as natural explanations given a general-purpose solver. Table 8.1 presents the statistics of the original models as well as of the diagnosis problems compiled to Horn using η. We observe in the table that the number of the hypotheses increases as well as the size of the theory which is directly connected to the number of non-Horn rules and their composition. The translation time with η took on average 68.5 milliseconds, however, the compilation to Horn is part of the off-line portion of the diagnosis process.

Table 8.1: $\mathfrak{L}$ and Horn model statistics.

		MIN	MAX	AVG	MED
$\mathfrak{L}$ Models	$\|Hyp\|$	6	301	129.0	112
	$\|A \setminus Hyp\|$	5	973	358.9	295
	$\|Th_{\mathfrak{L}}\|$	11	391	172.1	151
	$\|Facts\|$	1	98	41.8	40
	$\|Conj\|$	0	132	51.7	45
	$\|Disj\|$	1	125	51.1	48
Horn Models	$\|Hyp'\|$	9	616	293.9	352
	$\|A' \setminus Hyp'\|$	5	973	358.9	295
	$\|Th\|$	20	972	475.5	567
	$\|Facts\|$	1	98	41.8	40
	$\|Conj\|$	0	0	0	0
	$\|Disj\|$	0	0	0	0

Table 8.2: Experimental results on 144 samples (1440 comparisons).

	clingo | $\mathfrak{L}'$ Models				ATMS | Horn Models			
	MIN	MAX	AVG	MED	MIN	MAX	AVG	MED
Total Time [in ms]	< 1	90,430	5,548.71	1160	< 1	137,772.5	3,022.3	4.0
Runtime [in ms]	< 1	90,430	5,548.71	1160	< 1	137,768.4	2,588.61	2.2
Time $\hat{\eta}$ [in ms]	-	-	-	-	< 1	61,960.3	433.7	1.44
$\|\Delta^S_{\mathfrak{L}}\|$	1	256	12.1	1	1	256	12.1	1
Single Fault $\Delta^S_{\mathfrak{L}}$	0	5	0.2	0	0	5	0.2	0
Double Fault $\Delta^S_{\mathfrak{L}}$	0	14	0.2	0	0	14	0.2	0
Triple Fault $\Delta^S_{\mathfrak{L}}$	0	36	0.7	0	0	36	0.7	0
$\|\Delta^S\|$	-	-	-	-	1	22,800	336.0	1
Single Fault Δ^S	-	-	-	-	0	0.1	0.2	0
Double Fault Δ^S	-	-	-	-	0	45	0.7	0
Triple Fault Δ^S	-	-	-	-	0	1521	15.6	0

We used two scenarios and compared them to determine the computational effort of our approach. On the one hand, we evaluated the diagnosis time on the compiled Horn models using an ATMS written in Java as well as a Java implementation of η and $\hat{\eta}$ to convert the models to their Horn representation and the solutions back to the original hypotheses space. On the other hand, we made some adaptations to the $\mathfrak{L}$ models to ensure intuitive diagnoses when exploiting general-purpose solvers. In particular, as shown in the previous section given a theory $\{\Phi \rightarrow \alpha \vee \beta\}$ and an observation α, Φ is not an explanation as $\{\Phi \rightarrow \alpha \vee \beta\} \cup \Phi \not\models \alpha$. Therefore, an abductive reasoning mechanism would not provide the explanations we receive based on our transformed model. In this regard, we construct models for a general-purpose solver, where the disjunctions on the right hand side are replaced by conjunctions. We refer to these models as $\mathfrak{L}'$ within our analysis[2]. In addition, when constructing our *PHCAP*s we ensure to only consider positive observations which guarantees that based on these adapted $\mathfrak{L}'$ models the explanations coincide with the reasonable diagnoses we seek to extract.

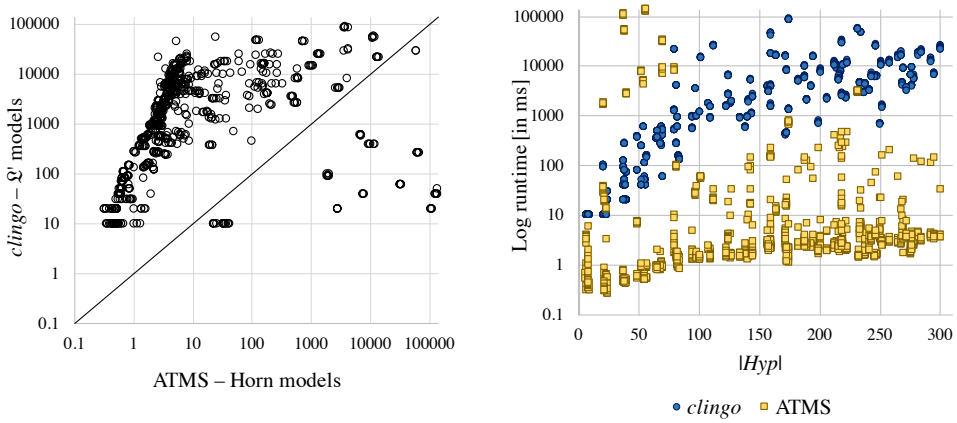
To compute the diagnoses on the samples written in $\mathfrak{L}'$ we exploit a logic programming meta-interpreter for propositional abduction and a logic program encoding of the diagnosis problems as proposed by Saikko, Wallner, and Järvisalo [Sai+16]. To account for all solutions we adapt their encoding by removing the included minimum-cost optimization statement. Yet, as mentioned, in order to yield intuitive explanations we do not directly convert the original $\mathfrak{L}$ system descriptions into disjunctive logic programs, but create normal logic programs, i.e., disjunctions on the right hand side are converted into conjunctions. We invoked the C++ ASP solver *clingo*[3] version 4.5.4 [Geb+14] to compute the abductive explanations based on the logic programs. As our interest is in deriving all subset minimal solutions we called *clingo* via the options `- -heuristic=Domain`, `- -dom-mod=5,16`, and `--enum-mod=domRec`, which invokes the enumeration of all subset minimal answer sets, i.e., all parsimonious abductive explanations.

For the experiments, each algorithm computed the minimal diagnoses for each sample ten times on a Mac Pro (Late 2013) with a 2.7 GHz 12-Core Intel Xeon ES processor and 64GB of RAM running OS X 10.10.5. Each diagnostic computation faced a 10 minute runtime limit and samples where one of the algorithms exceeded the timeout were disregarded to not distort the runtime report on $\hat{\eta}$ and thus the overall evaluation.

Table 8.2 presents the runtime results. The total computation time of our approach includes the diagnosis using the ATMS and the time to convert the results back via $\hat{\eta}$. For *clingo* we only considered its own measured execution time[4] as part of the entire runtime. Unsurprisingly, on average the total computation time on the Horn theories using the ATMS is faster than the ASP solver on the $\mathfrak{L}'$ models. Figure 8.2a graphically shows the execution time differences between the two approaches. Each data point depicts the relation between the log runtimes in milliseconds of one *clingo* execution to the total execution time of the compiled approach for a single diagnosis problem. Points above the diagonal represent sample runs, where diagnosis on the Horn representation was faster than the ASP solver. As can be seen, for the bulk of *PHCAP*s the ATMS computation is more efficient. However, there are certain samples which cannot be solved efficiently using the ATMS and/or require additional time for conversion to the original hypotheses space as can be seen from the corresponding maximum values in Table 8.2. Figure 8.2b illustrates for each *PHCAP* the runtime in relation to the number of hypotheses included within the original $\mathfrak{L}$ theory for both methods[5]. As can be seen there are two runtime behaviors for the ATMS; one

[2] In comparison to the system descriptions transformed via η, these $\mathfrak{L}$ models contain fewer clauses as well as hypotheses; hence, the resulting diagnosis problems are simpler than the one's based on the compiled theories.
[3] `https://potassco.org/clingo/`
[4] *clingo* reports its runtime as seconds, thus, for our analysis we converted them to milliseconds.
[5] Note here that due to the conversion of the execution reported by *clingo* in milliseconds, several sample runs feature the same computation time. Meaning that data points for *clingo* overlap in Figure 8.2b.

where the runtime only gradually increases with the number of hypotheses and one where the computation time grows more steeply. Investigating the reasons for this observation, we determine that the models with an extensive computation time feature less effects and a greater overlap on average, that is, the labels propagated by the ATMS are larger, require more time for updating and due to the more interconnected structure more label minimizations are likely to occur. Further, these *PHCAP*s are connected to a larger diagnosis sets Δ^S, which require more subset checks when mapped back with $\hat{\eta}$ since these promote an increased minimization time via μ.



(a) Scatter plot of total log computation times [in ms].

(b) Scatter plot of log runtimes [in ms] per number of hypotheses in the $\mathfrak{L}$ models.

Figure 8.2: Experiment results for the extended modeling.

8.4 Physics of Failure

As mentioned in the previous sections, the need for allowing a more expressive description of effect relations comes directly from industry, where observations alternatives may occur. Since a main focus of our work is to enable a seamless integration of model-based diagnosis in current work processes, we show how a failure assessment may be automatically used for constructing a knowledge base. In this regard we focus on the PoF method. The PoF approach to reliability analyzes root causes of failures and utilizes theoretical models capturing the relation between operational and environmental loads as well as damage accumulation rates. Thus, these mathematical models have to represent the physics inherent to the damage process. Based on the knowledge on failure mechanisms and potential degradation in addition to life cycle stress information, estimates can be made in regard to the probability of certain failures for a component or product under investigation [PD95].

Gray et al. [GW10; Gra+11] exploit this PoF approach for combining diagnostic and prognostic techniques to create an FMA. Their method relies on a structured evaluation of faults, in particular, physical and chemical failure mechanisms, and their manifestations. In order to create a comprehensive assessment of a system, it is decomposed into its subcomponents. Each part is then analyzed in regard to its potential failure modes as well as the damage driving physics. To ensure a complete depiction of the system, top-down and bottom-up reasoning are incorporated to determine failure relevant operational and environmental boundary conditions. For each combination of failure mode and component, a mathematical model is used to calculate the rate at which damage accumulates in response to the operating environment. The approach is similar to Failure Modes, Mechanisms and Effects Analysis (FMMEA) [Gan+05] as it emphasizes the evaluation of damage driving

conditions. The FMA method differentiates between automatically retrieved state indicators such as SCADA results and measurements from inspections. In contrast to FMEA or FMMEA, the evaluation considers a more detailed view on the relationships between failure effects. By incorporating logical connections, namely AND as well as OR, the description of effects is a Boolean expression representing which consequences are observed in case of a fault and how these are linked.

Example 8.4

Table 8.3 shows parts of an FMA produced by an expert group during a system analysis for the converter of an industrial wind turbine which we have seen in an adapted form in Example 7.1. As can be seen the analysis encompasses the failure mechanism, i.e. fault mode, the component, the operation and environmental conditions promoting the damage, as well as the effects which are on the one hand state indicators automatically retrieved from a condition monitoring system as well as manual part inspections.

Table 8.3: FMA Example.

Fault Mode	Component	Damage Promoting Operating Mode	Aggravating Boundary Conditions	State Indicators/Part Inspection
Electrical chemical aging	Buck Boost - Electrolyte Capacitor	Partial load	High ambient temperature	(T_power_cabinet *OR* P_turbine) *AND* ESR
Corrosion	Fan - Pin	Full load	Saline environment, high temperature environment	T_cabinet *OR* P_turbine
Thermo-mechanical fatigue (TMF)	Fan - Bearing Running Surface	Start Up/ Shut Down, Transient current/voltage events	Light winds (frequent start up and shot down), changing wind direction (yaw adjustment)	T_cabinet *AND* P_turbine *AND* Alarm-code_overvoltage_link
High-cycle fatigue (HCF)	IGBT - Wire Bonding	Start Up/ Shut Down, Transient current/voltage events	Low ambient temperature	T_inverter_cabinet *OR* T_nacelle *OR* P_turbine

8.4.1 Model Development

As we are creating propositional Horn clauses to form a *PHCAP*, some adjustments to the information stored in the FMA have to made. Again we can use three sources of information for modeling: the set of components *COMP*, their potential fault modes *MODES*, and the Boolean formulas *FORM* describing the connection between the state indicators and part inspections.

Definition 8.1 (Failure Mode Assessment (FMA)). An FMA is a set of tuples (C, FM, φ) where $C \in COMP$ is a component, $FM \in MODES$ is a fault mode, and $\varphi \in FORM$ is the Boolean expression relating effects to one another.

Since the effects can be connected by disjunctions, a simple mapping to Horn clauses as with the FMEA is not possible. We have to apply the mapping we have formulated in the previous section. Here, we give a presentation of the approach in terms of an FMA failure record. As a disjunction of effects implies that at least one of these manifestations is present in case of the failure, a mapping to conjunctions, as Horn clauses would require,

is inadequate. Thus, the underlying information has to be preprocessed in a new FMA which we denote FMA$_p$. The first step is to ensure that each Boolean expression in the effect description is in DNF, i.e., each formula is a disjunction of conjunctions. Basically, this ensures that the clauses we retrieve when mapping the record correspond to our language $\mathfrak{L}$. This conversion can be simply achieved by applying the laws of Boolean algebra. Let us assume that there is a function $\mathfrak{D}$ converting a Boolean expression ϕ into its DNF form:

$$\forall \varphi \in \mathit{FORM} : \varphi' = \mathfrak{D}(\varphi) \tag{8.7}$$

Thus, each tuple $(C, \mathit{FM}, \varphi') \in \mathit{FMA}_p$ now contains the DNF formula φ' of φ.

Now, we can apply the apply the mapping function η, thus, in the second step we create for each record, where φ' consists of a disjunction, a new fault mode FM_c for each conjunction $c \in \varphi'$. Each resulting tuple (C, FM_c, c) is added to FMA$_p$. It is apparent that each original tuple $(C, \mathit{FM}, \varphi')$ has to be removed subsequently from FMA$_p$.

Example 8.4 (cont.)

The first entry of the FMA in Table 8.3 results in the following tuple:

$$(\textit{ Buck\_Boost, Electrical\_chemical\_aging,}(\textit{T\_power\_cabinet} \vee \textit{P\_turbine}) \wedge \textit{ESR})$$

For the first entry, we can record φ' as

$$(\textit{T\_power\_cabinet} \wedge \textit{ESR}) \vee (\textit{P\_turbine} \wedge \textit{ESR}) \quad .$$

The Boolean expression in DNF of the first entry comprises a disjunction with two conjunctions, i.e. $(\textit{T\_power\_cabinet} \wedge \textit{ESR})$ and $(\textit{P\_turbine} \wedge \textit{ESR})$. Thus, for each of the conjunctions a new fault mode FM_c is generated and the resulting tuples of the form (C, FM_c, c) are:

$$(\textit{Buck\_Boost, Electrical\_chemical\_aging\_1}, \textit{T\_power\_cabinet} \wedge \textit{ESR})$$

and

$$(\textit{Buck\_Boost, Electrical\_chemical\_aging\_2}, \textit{P\_turbine} \wedge \textit{ESR}) \ .$$

It is apparent that the number of records within the FMA$_p$ is increased in regard to the primary FMA, as each DNF formula can be exponentially larger than its original[a]. Table 8.4 shows the processed FMA$_p$.

Table 8.4: Processed Failure Mode Assessment (FMA$_p$) of the Converter.

Fault Mode	Component	Damage Promoting Operating Mode	Aggravating Boundary Conditions	State Indicators/Part Inspection
Electrical chemical aging 1	Buck Boost - Electrolyte Capacitor	Partial load	High ambient temperature	T_power_cabinet *AND* ESR
Electrical chemical aging 2	Buck Boost - Electrolyte Capacitor	Partial load	High ambient temperature	P_turbine *AND* ESR
Corrosion 1	Fan - Pin	Full load	Saline environment, high temperature environment	T_cabinet
Corrosion 2	Fan - Pin	Full load	Saline environment, high temperature environment	P_turbine

Thermo-mechanical fatigue (TMF)	Fan - Bearing Running Surface	Start Up/ Shut Down, Transient current/voltage events	Light winds (frequent start up and shot down), changing wind direction (yaw adjustment)	T_cabinet *AND* P_turbine
High-cycle fatigue (HCF) 1	IGBT - Wire Bonding	Start Up/ Shut Down, Transient current/voltage events	Low ambient temperature	T_inverter_cabinet
High-cycle fatigue (HCF) 2	IGBT - Wire Bonding	Start Up/ Shut Down, Transient current/voltage events	Low ambient temperature	T_nacelle
High-cycle fatigue (HCF) 3	IGBT - Wire Bonding	Start Up/ Shut Down, Transient current/voltage events	Low ambient temperature	P_turbine

[a]Note here that in our modeling approach in Section 8.2, we already restrict the right hand side of disjunctive extension in $\mathfrak{L}$ to be in DNF.

Then, the mapping $\mathfrak{M}_{FMA} : 2^{FMA_p} \mapsto HC$ is similar to $\mathfrak{M}$ and the composition of *Hyp* (Equation (8.10)) and *A* (Eq. (8.11)) is analog to the FMEA modeling.

Definition 8.2. Given an FMA$_p$, the function $\mathfrak{M}_{FMA}$ is defined as follows:

$$\mathfrak{M}_{FMA}(FMA_p) =_{def} \bigcup_{t \in FMA_p} \mathfrak{M}(t) \tag{8.8}$$

where

$$\mathfrak{M}_{FMA}(C, FM_c, c) =_{def} \{mode(C, FM_c) \rightarrow v \,|\, v \in c\} \tag{8.9}$$

$$Hyp =_{def} \bigcup_{(C,FM_c,c) \in FMA_p} \{mode(C, FM_c)\} \tag{8.10}$$

$$A =_{def} \bigcup_{(C,FM_c,c) \in FMA_p} \{mode(C, FM_c)\} \cup \bigcup_{v \in c} v \tag{8.11}$$

Example 8.4 (cont.)

Applying Equation (8.10), Equation (8.11), and $\mathfrak{M}_{FMA}$ to the FMA$_p$ in Table 8.4 results in the following *KB*:

$$Hyp = \left\{ \begin{array}{c} mode(Buck\_Boost,\ Electrical\_chemical\_aging\_1), \\ mode(Buck\_Boost,\ Electrical\_chemical\_aging\_2), \\ mode(Fan\_Pin,Corrosion\_1), \ldots \end{array} \right\}$$

$$A = \left\{ \; mode(Fan\_Pin,Corrosion\_1), T\_power\_cabinet,\ P\_turbine, \ldots \; \right\}$$

$$Th = \left\{ \begin{array}{c} mode(Buck\_Boost, Electrical\_chemical\_aging\_1) \rightarrow T\_power\_cabinet, \\ mode(Buck\_Boost, Electrical\_chemical\_aging\_1) \rightarrow Equivalent\_series\_resistance\_higher, \\ mode(Buck\_Boost, Electrical\_chemical\_aging\_2) \rightarrow P\_turbine, \\ \ldots \end{array} \right\}$$

From Section 8.2, we know that the computed diagnoses are not equivalent to the results on the original model and the solutions do not contain the original fault modes, but the additionally created ones of the second transformation step. Thus, these have to be mapped back once the explanations have been computed. Also recall that due to the creation of the auxiliary hypotheses the abductive solutions when converted back to the original fault modes might not be minimal and it is necessary to ensure that the solutions are consistent with the background knowledge.

8.4.2 Advantages and Limitations of Using PoF

This approach combines prognosis and diagnosis by determining damage models for recorded fault modes. As FMA allows to express the combination of effects with disjunctions a simple mapping to a Horn theory is not possible. Thus, a conversion prior to model creation is necessary. The disadvantage is that the resulting processed model might be exponentially larger than the original assessment due to the transformation, that the resulting diagnoses have to be mapped back to the initial causes and that finally subset checks have to be performed to ensure minimal explanations. Furthermore, due to the inclusion of disjunctions between effects, the discrimination capability of probing is limited to a certain extend. The main benefit of the FMA is the prognosis capabilities based on the knowledge of failure mechanisms and life cycle stress. In comparison to the other failure assessments we have considered as a basis for an automatic modeling approach in Section 3.3, i.e., FMEAs and fault trees, the incorporation of information of time to failure, allows a more accurate ranking of diagnoses.

8.5 Conclusions

In this chapter, we have shown a conversion from a subset of propositional logic to Horn sentences in the context of abductive model-based diagnosis. By allowing a more expressive representation of the system to be diagnosed, a more accurate depiction of the failure behavior can be encoded and intuitive explanations can be derived. We do not claim the diagnoses are equivalent to the ones on the original model. Nevertheless, we argue that the solutions generated are more desirable from a practical point of view as they resemble human diagnostic reasoning. In particular, we have defined an extension of Horn clauses, which may feature a disjunction at the head of a rule, allowing to specify effects which not necessarily all be observable given a failure. The need for this has been raised by industry. Differentiating between system manifestations occurring in conjunction and symptom alternatives is essential in order to accurately specify failure behavior and not prematurely rule out diagnosis candidates. Yet, the transformation functions we have presented are straight forward, easy to implement, and allow the usage of any Horn clause abductive reasoning system to receive these natural explanations.

To convey the practicality of the approach we have conducted an initial empirical experiment using artificially generated samples where we compared the computation time of our approach, including the diagnosis on the compiled models and the transformation back to the original hypotheses space, to abduction using a logic program encoding with some

adaptations on an ASP solver. From the data we conclude that even though the size of the model increases due to generation of additional hypotheses, the diagnosis on the compiled model using a Horn theorem prover is more runtime efficient than the computation based on an ASP solver. While this observation is not surprising, the main benefit from using our model compilation is the resulting intuitive diagnoses. Further, the transformation allows exploiting other solvers specifically tailored to abductive reasoning which are limited to Horn even when the underlying model does not entirely conform to this restriction.

Lastly, we have considered the PoF approach to reliability analysis that combines prognosis and diagnosis by determining damage models for recorded fault modes. The failure assessments created by our industrial partner based on the PoF method allow to express the combination of effects with disjunctions such that a simple mapping to a Horn theory is not possible. In addition, diagnosis on the original model would not lead to the diagnose we aim to generate in a practical context. Thus, a compilation of the knowledge to Horn is necessary. The disadvantage is that the resulting processed model might be exponentially larger than the original assessment due to the transformation[6], that the resulting diagnoses have to be mapped back to the initial causes, and that finally checks have to be performed to ensure minimal consistent explanations. The main benefit of the PoF approach is the prognosis capabilities based on the knowledge of failure mechanisms and life cycle stress. In comparison to FMEA and FTA, the incorporation of information of time to failure, allows a more accurate ranking of diagnoses.

What is missing is the semantic operation necessary to define these more rational explanations in a logical framework. It is clear that there is not an entailment relation between the diagnoses and the observations w.r.t the background theory, but the explanations are consistent with the theories. Other future work may include an extension of the source language, a more detailed investigation of the trade-offs between runtime and expressiveness, and additional evaluations of the approach. We are planning on assessing the technique in the wind turbine domain. There our industrial partner and one of their customers are currently operating and evaluating an implementation of our method. This trial should give us insights into whether the approach can be applied successfully to an industrial application in regard to computation times and fault identification effectiveness. We give some insights in the realization in the upcoming chapter. Additionally, an analysis on a benchmark featuring practical samples would aid in conveying the feasibility of the compilation and its resulting diagnoses.

[6]This exponential explosion depends on the Boolean expression within the failure assessment and not on the mapping function we have presented in Section 8.2.

9 Designing a Diagnosis Application and its Graphical User Interface

> *It is the pervading law of all things organic and inorganic, of all things physical and metaphysical, of all things human and all things superhuman, of all true manifestations of the head, of the heart, of the soul, that the life is recognizable in its expression, that form ever follows function. This is the law.*
>
> **— Louis Sullivan**
> "*The Tall Office Building Artistically Considered*". 1896.

This chapter is based on the following publications:

- [Koi+17] Roxane Koitz et al. „Model-Based Diagnosis in Practice: Interaction Design of an Integrated Diagnosis Application for Industrial Wind Turbines." In: *Proceedings of the 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*. 2017, pp. 440–445
- [Koi+18] Roxane Koitz et al. „Wind Turbine Fault Localization: A Practical Application of Model-Based Diagnosis". In: *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*. Cham: Springer International Publishing, 2018, pp. 17–43. ISBN: 978-3-319-74962-4. URL: https://doi.org/10.1007/978-3-319-74962-4_2

This chapter is based on [Koi+18] of which a preliminary version has been published in [Koi+17].

9.1 Motivation

As already mentioned, the dissemination of model-based diagnosis in practice is still insignificant. While we have already addressed the issues of (1) developing diagnosis models [CD99] by defining a process for abductive diagnosis in industrial practice that is based on automatically extracting the necessary knowledge from records available and (2) the computational effort of computing explanations through the evaluations we have conducted, we have not discussed a real application so far. Although the previous two chapters report on the initial situation and modeling challenge, in this chapter we attend to the consolidation of model-based diagnosis and current industrial work processes, which is an essential necessity for integrating new technology [Mil+00].

In cooperation between Graz University of Technology and Uptime Engineering the AMOR project was initiated with the aim of providing a methodology and framework for model-based diagnosis in the industrial wind turbine domain. Due to Uptime Engineerings many years of experience and expertise in the field of wind power plant maintenance, industrial wind turbines constitute an ideal test bed and application area for model-based diagnosis. In cooperation with Uptime Engineering, the self maintenance of wind power plants of a local Austrian wind energy provider should be facilitated and improved in regard to efficiency and effectiveness. Usually, wind turbine maintenance is based on a service contract with

the turbine manufacturer. In case of a failure, the maintenance provider schedules the repair work without consideration of current wind conditions or already schedule planned maintenance activities. Given the competitive nature of the wind energy sector this can have a negative effect on the efficiency of the wind parks and hence affect the revenue of the energy provider. Moreover, O&M costs are not evenly distributed over the turbine life time but become more extensive as time progresses due to an increased component failure probability and limited manufacturer warranty periods.

By maintaining their wind turbine installations themselves, the energy provider aims at increasing the efficiency and effectiveness of the repair and replacement undertakings. In particular,

- non-critical repairs should be scheduled with planned maintenance tasks,
- turbine downtime should be minimized,
- by scheduling maintenance assignments in consideration of the wind forecast the energy harvest should be maximized,
- repair of broken components should be promoted in contrast to replacement of parts, and
- by equipping the service technicians with the correct tools and required parts for maintenance activities to travel efforts and hence maintenance time should be reduced.

Currently, the troubleshooting of the installations is mostly based on the SCADA alarm codes reported by the wind turbine. However, as there is no direct correspondence between failures and alarm codes and the codes do not provide information on the faulty component, diagnosis is mainly based on the experience of the service technicians [Lüf18].

In this chapter, we present some of the results of the collaboration as well as the ongoing realization of model-based diagnosis in practice. With this work we aim to bridge the gap between the theory of model-based diagnosis and its practical application. In particular, we tackle two barriers identified by Travé-Massuyès and Milne [TMM98], namely we focus on human-computer interaction, i.e., how does the operator interact with the model-based diagnosis system and which parameters do we use to increase the likelihood of system acceptance and use even after the project is finished. Furthermore, we discuss how we consolidate our model-based diagnosis tools and the existing condition monitoring software in our project. We introduce an application designed to facilitate troubleshooting in the industrial wind turbine domain. In particular, the application's GUI is presented, which has been created taking the needs, current work processes, and environments of the maintenance personnel into consideration and aims at enhancing the effectiveness and efficiency of the maintenance task. To ensure a usable interface and functionality, an iterative design process was applied that incorporates continuous feedback from several stakeholders. Subsequently, we discuss the current status of the integration of the model-based fault identification engine in the industrial wind turbine domain. In the end, the diagnosis application should be part of an existing monitoring system. Lastly, we give concluding remarks.

9.2 Abductive Model-Based Diagnosis Prototype

To enable model-based diagnosis in industrial practice as proposed in Section 3, the necessary failure information must be available to automatically extract an appropriate diagnostic model and an anomaly detection method is needed to initiate the fault identification phase. Furthermore, in order to yield benefits from deploying such a system, solutions need to be

computed efficiently[1] and effectively reflecting defects present in the system. We argue, however, that these technical features are not the only deciding factors determining the success of a newly integrated diagnosis software. While current research frequently focuses on developing and improving reasoning techniques, the suitable integration of model-based diagnosis in operational processes is rarely addressed [Mil+00]. In addition, it is well known that the acceptance of new technology is tightly linked to the perceived usefulness of the product as well as its perceived ease of use [Dav89]. The former refers to the benefits for the users and other stakeholders in regard to the performance of work tasks, whereas the latter is on par with the usability[2] of a product.

Hence, in developing a model-based diagnosis application for use in the field within our project, we focus not only on the technical aspects of feasibility but further account for the human factor. An interface and interaction design was incrementally developed for an abductive model-based diagnosis engine that should function as a template for the actual implementation of the tools that is currently being integrated into Uptime Engineering's software. Various prototypes were created iteratively, starting from a low-fidelity paper mock-up to a clickable prototype depicting a usual fault identification scenario. These prototypes reflect the above-described general process of abductive model-based diagnosis in the context of wind power plants. Particular attention was paid to respecting current work processes and accounting for a usable design.

The design process started with eliciting the requirements of the diagnosis application in consideration of the stakeholders involved in the project, who are:

- the service technicians, who are the users, will operate the diagnosis software for troubleshooting from the service center and also in the field, and are responsible for performing the turbines' planned maintenance, repair as well as replacement activities
- the management of a wind energy provider, planning on extending their self-maintenance activities for their wind turbine plants in the future
- Uptime Engineering, who currently develop condition monitoring software for wind turbines and seek to extend their portfolio with usable and extendable diagnosis software

Requirements

A list of requirements in regard to the final diagnosis application was established during the course of the various design iterations. The three distinct stakeholder groups have differing requests, which were analyzed in order to resolve conflicts and prioritize the resulting requirements. Since the success of the application depends to a great extent to being used by the service technicians, special attention was given to their suggestions and needs.

An important observation is that current fault detection activities performed by the service personnel typically rely on visual inspection. Hence, in order to support diagnosis, images should be used for easier recognition. Once a fault has been identified, the repair or replacement task is executed according to the wind turbine manufacturer's instruction manuals. Therefore, such documents need to be easily accessible via the software. After the maintenance activities have been completed, the service technicians are required to create a report of the assignment and actions performed. The software should thus support

[1] Here, efficiency is subjective to the application domain, e.g., in the context of wind turbines deriving explanations in minutes or even hours is sufficient, while for automotive on-board diagnosis this computation time is unacceptable.

[2] Usability is defined in the ISO standard 9241 part 11 [Sta10] as *"Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."*

automation of the reporting step to reduce the overall effort. As the working environment inside a wind turbine is often uncomfortable and limited in space, and work is performed under time pressure in potentially difficult weather conditions, the user interface therefore needs to be intuitive in use and must guide the user through a strictly defined sequence with minimal user interactions. Considering the overall work process, the software should feature a desktop software part operated in the service center as well as a mobile application, which should be used within the turbine itself.

The management of the energy provider is interested in promoting digitalization as well as increasing the productivity and safety of their wind operations. On the one hand the software should support the service technicians in preparing all spare parts and tools necessary before traveling to a wind turbine to ensure minimal downtime, while on the other hand given the hazardous environment in the field it should support the safety processes already in place, e.g., the service technicians personal safety equipment. In addition to the user and management requirements, the specifications of Uptime Engineering needed to be satisfied. To extend and update the knowledge base, i.e., abductive model, the users should be able to report new fault modes, which have not been previously contemplated. Further, the user interface should be extendable and adaptable to satisfy other customers as well as other domains for future projects.

Design Process

To ensure a user friendly end product, the diagnosis engine's GUI was developed using an iterative process [Nie93]. Each iteration starts with a definition or adaptation of the requirements, then a design is created and subsequently a prototype is implemented. This prototype is evaluated by users from the target group to determine usability issues, which must be fixed in the interface and interaction of the proceeding cycle. According to Nielsen [Nie93] due to the various repetitions of this loop, this type of design methodology allows gaining sufficient insight into usability issues even given a limited number of test users.

In our case, the first iteration was kicked-off with a meeting between Graz University of Technology and Uptime Engineering to elicit the first set of requirements. One of the main goals identified was that the software should be designed in a way that supports the service technicians' current work processes without causing additional effort. Facilitating the service personnel's work tasks is essential as this assures usefulness, which is a key aspect in technology acceptance [Dav89]. Furthermore, we defined the overall workflow for the application, the general structure for an initial paper mock-up, and a small set of features, which should be realized. The initial paper prototype and all following designs were evaluated at meetings with the management of the energy provider and service technicians. At these meetings the current prototype was presented and a more detailed knowledge about the users and their work process was gained, usability issues could be uncovered and useful features that would aid the maintenance personnel throughout the fault correction process, were identified. During the first iterations predominant usability problems were detected and more drastic changes to the design were introduced, while in the later cycles only minor issues were found and as a result only slight GUI alternations were necessary.

The product of the design phase is a clickable prototype which has undergone a small-scale qualitative usability test involving five service technicians. In the test scenario, the participants were asked to use the clickable prototype to a mock-up fault identification process from start to finish, i.e., from the detection of an abnormal behavior by the condition monitoring system over performing the diagnosis and reporting the planned/conducted repair or replacement while articulating their thoughts and impressions on the system. Based on the video material taken during the evaluation a set of usability issues where identified

and using questionnaires and interviews a general notion of the usability of the interface was obtained. Although the service technicians were part of the elicitation of requirements and the design, several problems were uncovered in the experiment. In addition, the interviews revealed that all operators have had negative experiences with diagnosis systems in the past and hence question the usefulness and real benefit of the application. The design of the final prototype is presented below together with the overall application workflow.

9.2.1 Workflow and GUI Design

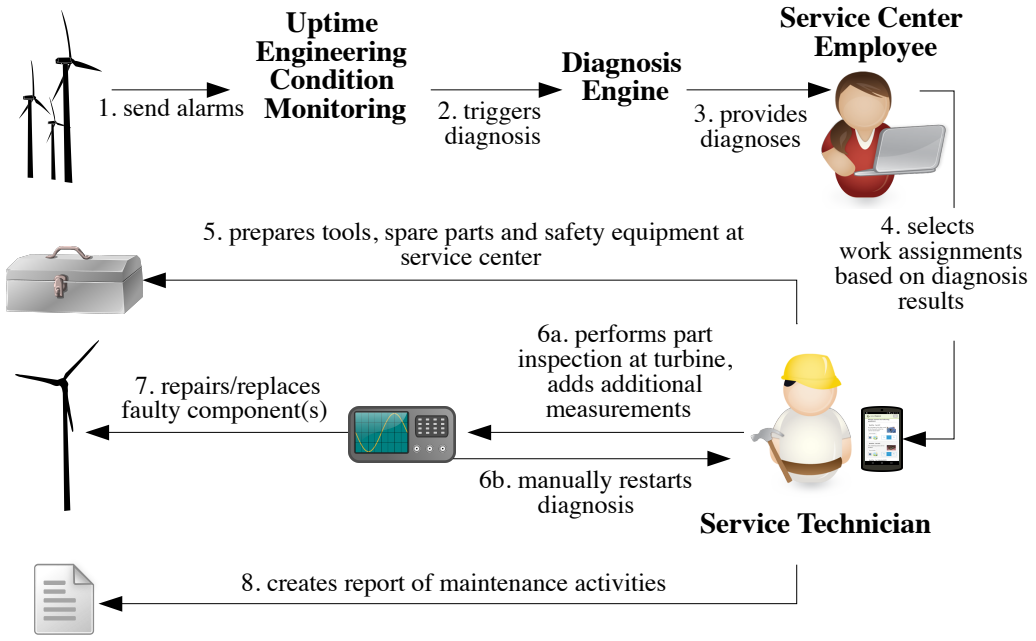


Figure 9.1: Workflow of the diagnosis application.

The workflow of the diagnosis application was created in consideration of the current functionality of Uptime Engineering's condition monitoring software, the maintenance process of the energy provider, and the general abductive diagnosis process of Chapter 3. Figure 9.1 depicts the identified activity sequence of the troubleshooting. The interface and interaction design decisions of the application were taken based on the workflow and requirement analysis. A fault identification is invoked once an anomaly has been encountered. Each wind turbine includes a set of sensors and a basic on-board system that triggers alarms whenever measurements fall outside certain limits (Step **1** in Figure 9.1). Uptime Engineering's condition monitoring software extends and refines the fault detection by further processing the available sensor information.

Once a symptom of a faulty turbine has been identified, Uptime Engineering's software triggers the diagnosis operation by supplying the previously created system description as well as the observations to a model-based diagnosis engine (Step **2** in Figure 9.1). After the computation, the results are accessible to the employees at the service center (Step **3** in Figure 9.1). The diagnoses are displayed per turbine instance and displayed as collapsible panels within Uptime Engineering's web interface, i.e., at the *Operations Center*, which is depicted in Figure 9.2 and designed for desktop or laptop computers. For each triggering

symptom, e.g., *Error Converter Bus*, the panel contains the possible root causes[3] as well as diagnosis likelihood expressed as percentages.

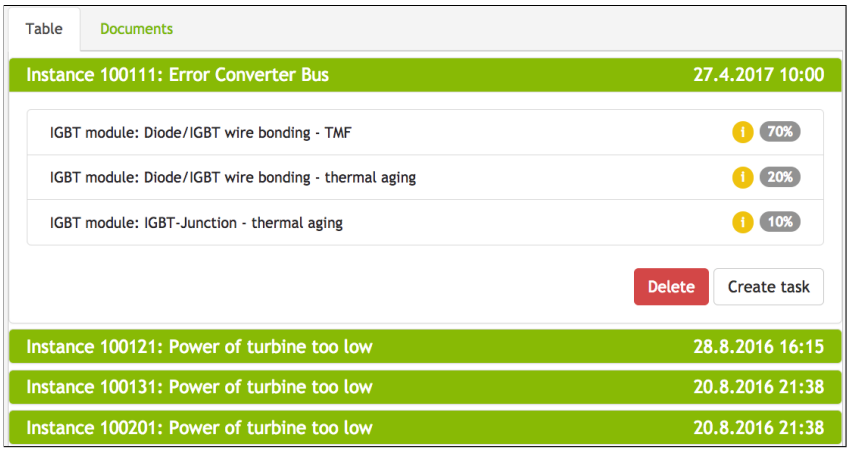


Figure 9.2: *Operations Center*.

Based on the outcome, the service center employee can create and assign repair tasks for the service technicians preselecting some of the possible faults for consideration during the field work (Step **4** in Figure 9.1). Each repair task is either preformed in conjunction with the next planned maintenance, scheduled, or immediately executed. Figure 9.3 depicts an example for a scheduled repair tasks, consisting of the anomaly and the corresponding error codes. The service center employee can then define a troubleshooting task, schedule the activity under consideration of the time table depicting the availability of service technicians, assigning both a supervisor and a team for the undertaking, add the corresponding parts and tools to the work assignment depending on the failures proposed by the engine, and provide additional auxiliary information to the maintenance job such as previous issues with the targeted wind turbine.

In the context of fault identification within the field, we concluded that the software would be most usable on a mobile device since the technicians prefer not to carry a laptop. Thus, once all work assignments for a team have been created, the rest of the diagnosis process is conducted by the service technicians over a mobile application. An essential aspect of the software design, was to follow guidelines and best practices for mobile user interfaces to ensure an easy to use application. A flat navigation was thus chosen for the prototype featuring little nesting of sub levels, and thus warranting minimal user interaction and proving a defined role in the work process of the technicians. A simple navigation drawer is used to allow the user to switch quickly between the top-level sites. In Figure 9.4a the *Home* screen of the mobile application is shown, where the technician can see all work assignments for the day as collapsible panels with additional information. Based on their tasks the technicians can obtain a list containing all necessary spare parts, tools, and safety equipment required for all maintenance activities planned on that day from the *Preparation* view depicted in Figure 9.4b. The preparation would usually be performed at the service center, where the stockroom is located (Step **5** in Figure 9.1). Once at the turbine, an overview of the maintenance task for this particular instance and assignment is shown in

[3] In the case of wind turbines, there is generally a strong single fault assumption. Thus, each depicted root cause in this example only consists of a single failure, e.g., *IGBT module: Diode/IGBT wire bonding - TMF*. Yet, the diagnosis engine is of course capable of determining multiple fault diagnoses.

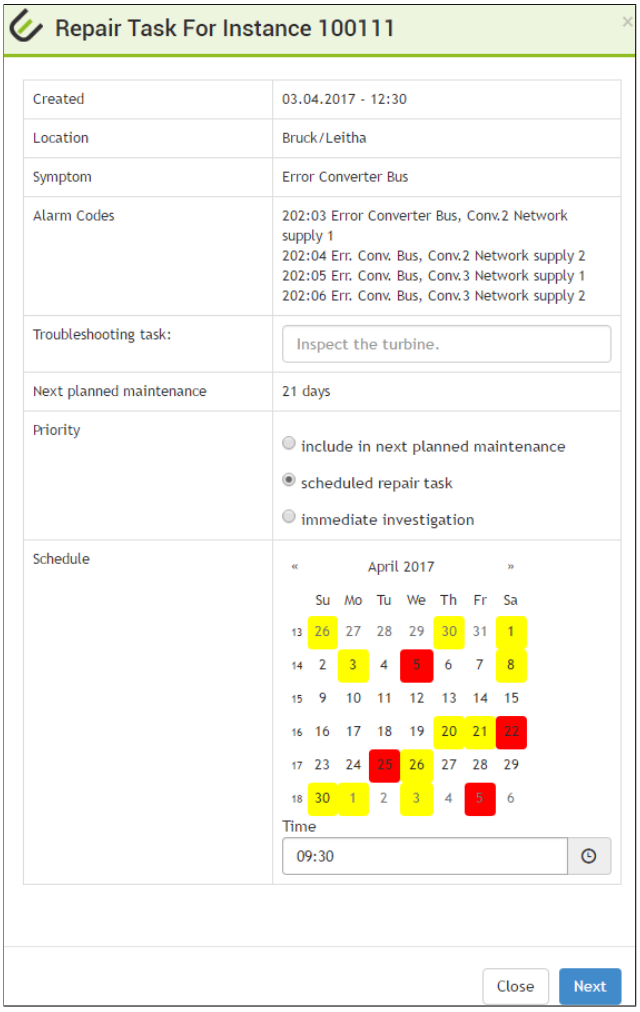
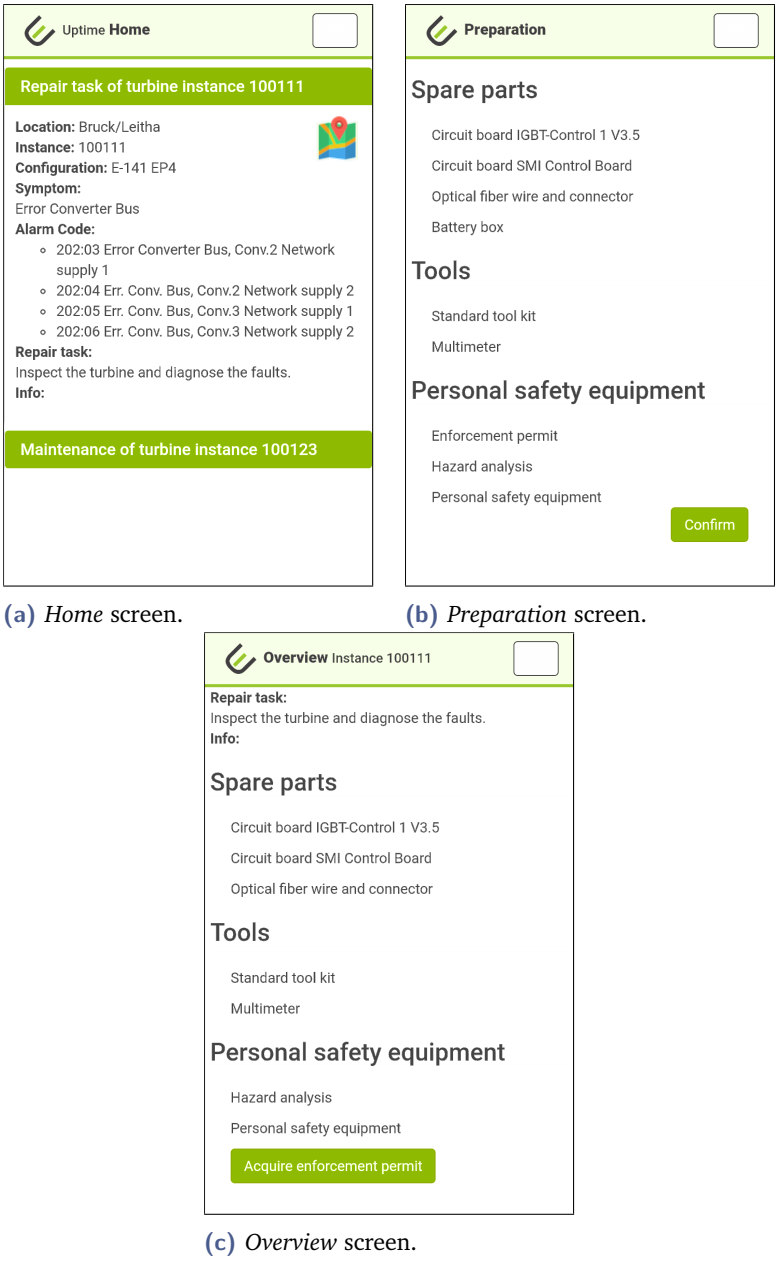


Figure 9.3: Repair Task Screen.

the overview screen (see Figure 9.4c), where an enforcement permit for the activity must be acquired[4].

When at the turbine, the service technicians can review the computed and ranked explanations represented in a collapsible panel as well as their percentage value (see Figure 9.5a). Expanding a diagnosis, a short description of the root cause of the failure is depicted as well as a **Repair task** button, leading to additional information on the required activities to restore a healthy turbine state, e.g., manufacturer manuals on repair and replacement of the faulty component. In cases where there is still no clear indication of the most likely diagnosis, the service technician can improve the initial results by supplying additional observations via **Improve diagnosis**. The next best probing points are determined via entropy as described previously. To obtain the additional turbine state information the application asks the users to answer a set of questions as seen in Figure 9.5b (Step **6a** in Figure 9.1). Besides the textual representation, a picture is provided for each probe to facilitate visual identification. Each observation can either be confirmed (**Yes**), denied (**No**), or bypassed (**?**). The device camera can be used to document the measurements separately and the pictures are later appended to the maintenance report. The layout to enter new symptoms ensures efficient use of available screen space, logically structures the tasks, and requires minimal

[4]A notification for the person responsible for the entire installation is automatically generated containing the request. Only after the permission has been granted, the technicians may perform the maintenance work.



(a) *Home* screen.

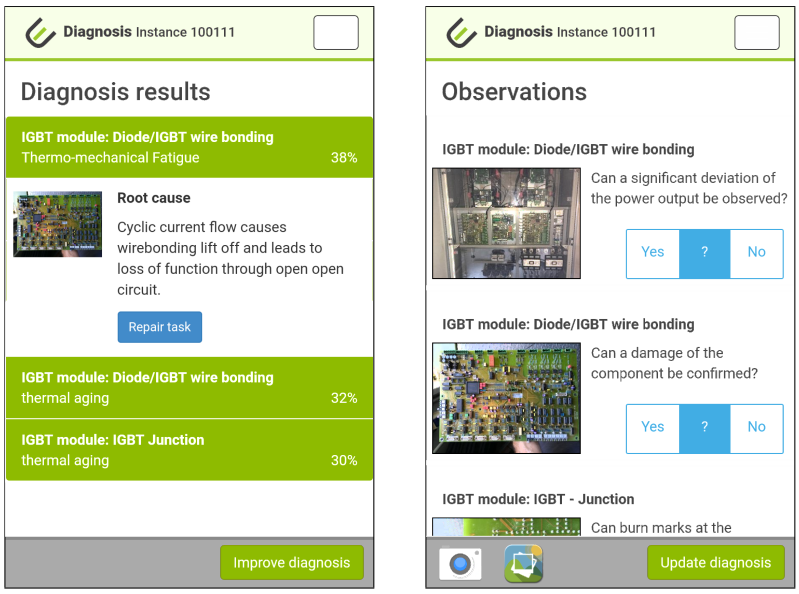
(b) *Preparation* screen.

(c) *Overview* screen.

Figure 9.4: Preparation and overview interface.

interaction. Once the additional observations have been made, the diagnoses are updated by manually restarting the computation given the new information from the device (Step **6b** in Figure 9.1). Given the updated results, the probabilities and arrangements of the faults change accordingly in the *Diagnosis* screen with arrows indicating if the fault converges or not (see Figure 9.5c). The diagnoses can be refined several times until an acceptable certainty for a fault has been reached.

Once the root cause of the detected anomaly has been repaired (Step **7** in Figure 9.1), the service technicians must create a report of the activities (Step **8** in Figure 9.1). This has previously been a tedious task and therefore an essential requirement by the users was that the tool provides support for their reporting. Thus, the mobile application allows the user to notify the system of the final confirmed diagnosis as well as spare parts consumed and repair or replacement activities carried out (see Figure 9.6a and 9.6b). The observations made



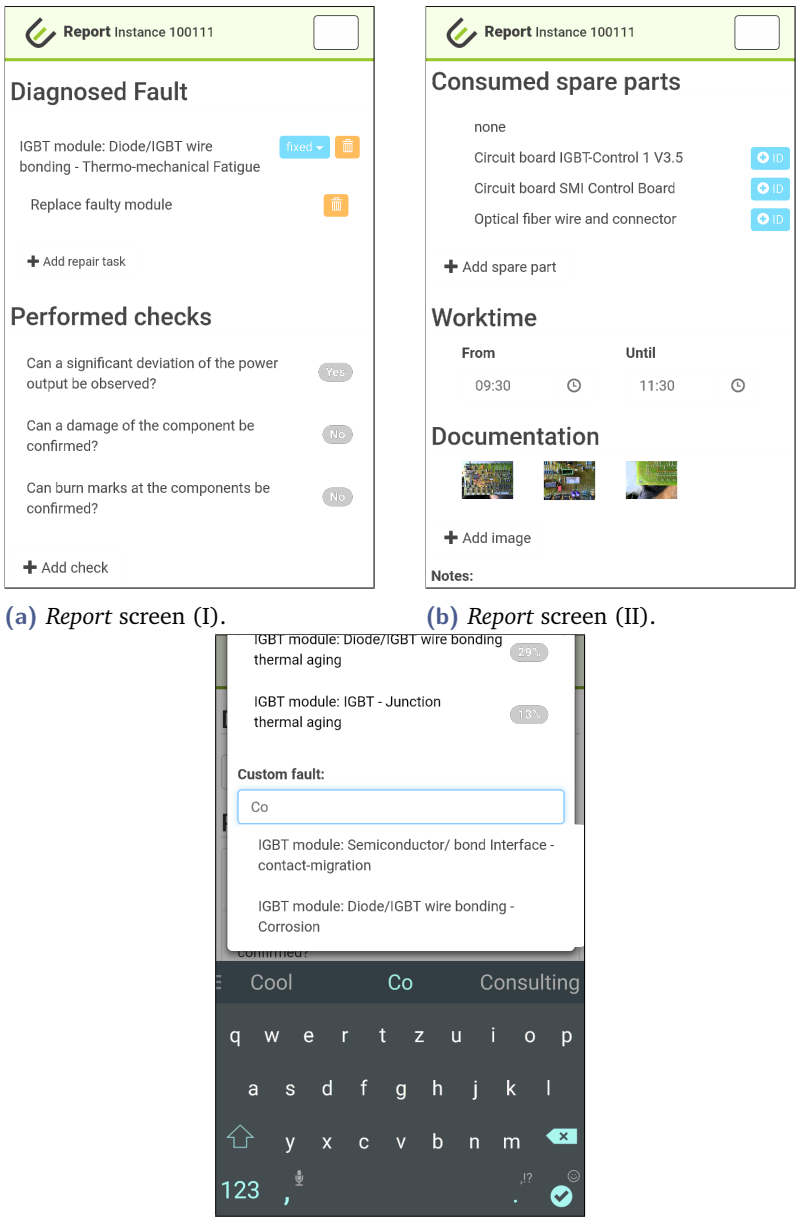
(a) Initial *Diagnosis* screen.

(b) Adding observations.

(c) *Diagnosis* screen after recomputation.

Figure 9.5: Diagnosis and probing interface.

and their visuals are automatically included in the report. Since it might be possible that a fault has not been considered within the failure assessment and thus abductive model, the interface provides a simple way for the service technician to reject all proposed diagnoses and add a custom fault, as shown in Figure 9.6c. Based on the entered data the work assignment documentation is automatically generated and sent to the *Operations Center*. There it is either stored or post processed, depending on whether crucial information is missing from the report or a custom failure has been created, which needs to be analyzed and subsequently added to the failure assessment. It is currently planned to have the knowledge acquired from the custom failures inserted into the failure assessment manually after ensuring the information provided is sound and complete.



(a) *Report* screen (I).

(b) *Report* screen (II).

(c) Adding a new custom fault.

Figure 9.6: Mobile reporting interface.

9.2.2 Realization of the Diagnosis and Modeling Engine

The full integration of the diagnosis engine as well as the associated user interfaces into the existing Uptime Engineering condition monitoring tool was dealt with in three main project phases. Initially it was necessary to introduce a structured taxonomy within the failure assessments and the communication with the diagnosis engine to allow an automatic processing. Next an interface was generated between the condition monitoring system and the model-based diagnosis engine, via messaging bus. Finally, the user interface of Uptime Engineering's software has to adapted to incorporate the functionality as described in the previous section. Each of these project phases is now described in further detail.

The use of a strict taxonomy to describe individual system components such as bearing, filter, fan as well as sub-systems such as gearbox, generator, pitch drive within the turbine

is critical for effective automation of the diagnostic process. The absence of such a system would lead to errors in the definition of observed system behavior, issues in regard to the model generation based on failure assessments, as also problems in interpretation of the recommendations. Furthermore, the naming of input signals (e.g. power, wind speed, oil temperature) and the behavior of such signals (e.g., "high power relative to the expectation, as defined by historical power performance"), must be coded according to a strict and consistent taxonomy. Once established, this taxonomy must be applied accurately and consistently throughout the software environment. Further complication results from the need to develop a general solution that can be deployed across multiple organizations (e.g., several different energy utilities) and monitor technology from a variety of wind turbine manufacturers, each with its own internal preferences for system taxonomy. Therefore much attention was paid to the generation of a global, independent taxonomy specifically suited to the purpose of anomaly detection and fault diagnosis.

The model-based diagnosis engine provided by Graz University of Technology is based on a Java implementation of the model mapping as well as an ATMS for the fault identification portion and the entropy-based computation of the next best probing point. There are two rationals for choosing the ATMS in this context. First, we have seen in Chapter 5 that the ATMS provides satisfactory computation results for Horn clause abduction in regard to efficiency. In addition, we know from the application domain that (1) diagnoses are to be computed in rather large intervals, e.g., once a day, and (2) the number of components considered within the FMA is limited to a certain extended. Second, based on the ATMS nodes' labels we can conveniently compute the next best observation.

Uptime Engineering's the condition monitoring system was developed using a range of technologies including .NET, MSSQL, Matlab, Java. The frameworks and languages have been selected based on their capabilities to perform signal analysis as well as their suitability for building commercial tools. An effective communication between these two systems was achieved using a Messaging Bus solution. Upon detection of abnormal system behavior, the condition monitoring software generates Events, with a specific content adhering to the taxonomy rules aforementioned. This Event is stored within the condition monitoring system and also converted into a message which is sent via the bus system to the reasoning engine. The message contains the latest observations for a specific wind turbine and the current failure assessment to be converted to an abductive *KB*. While the model generation can be performed offline, we decided to perform the transformation of the failure analysis whenever a diagnosis is triggered since the mapping implementation is very efficient and thus constitutes a negligible factor in the overall computation time. In addition, this means the engine can be memory-less from one diagnosis computation to the next and failure assessment updates are always considered during fault identification. The model-based reasoning engine then processes the information received and generates its own message, containing all possible root causes as well as recommendations for additional turbine state information that could be added as effective discriminators and hence improve the accuracy of the diagnosis. Note here that for each turbine instance a new model-based diagnosis engine instance is created once an abnormal behavior has been encountered. Through this exchange of information, the capability of the condition monitoring system is significantly enhanced.

An intuitive and efficient user interface is seen as a critical factor for acceptance of the system by its users, as described in Section 9.2 above. Since the software application is used both in the office environment at the wind turbine service center and on mobile applications by service staff working on the wind turbines, it was important to develop a graphical interface that works effectively in both cases. In order to provide the user with a consistent solution and also to minimize development effort, it was decided to create "adaptive user

interfaces", which recognize the device currently in use and automatically switch to the most suitable layout. The diagnosis GUI is currently in a prototype stage and still in development, while the back-end has already been fully developed and tested. Parts of the system, however, are already being evaluated by Uptime Engineering and the energy provider. Due to the lack of a user interface at this point in time, a broad and exhaustive acceptance study is planned for future work. The final product is expected to provide all relevant stakeholders with a comfortable user experience and improve and facilitate the fault diagnosis procedure of the wind turbine operator and its maintenance staff.

9.3 Conclusion

The theories and techniques of model-based reasoning have been applied to industrial problems, but the approach is not well represented in practical every-day use. We applied the process defined in Chapter 3 to the domain of industrial wind turbines. Uptime Engineering maintains a comprehensive failure analysis of turbine faults and their effects, which can be used as input for the automatic diagnosis model generation. Furthermore, their condition monitoring software allows the detection of abnormal turbine behavior. Graz University of Technology implemented an automated modeling tool creating abductive knowledge bases from failure analyses available as well as a model-based diagnosis engine that computes explanations based on the determined symptoms and provides recommendations on additional observations to refine the diagnosis results based on entropy.

In this chapter, we show how we overcome barriers of integrating model-based diagnosis: the main focus is on designing the human-computer interaction of a diagnosis application in the industrial wind turbine domain and in addition we describe how we join our diagnosis engine and the existing condition monitoring software. Various aspects, e.g., such as ensuring an appropriate communication between existing software tools, need to be considered when deploying model-based diagnosis to practical domains as the wind turbine industry. Moreover, (1) integrating the approach into already established work processes, (2) delivering a benefit to the users of the diagnosis software, and (3) providing intuitive and easy-to-use interfaces and interactions are key factors in promoting user acceptance. In this regard, a main goal was to facilitate not only the diagnosis portion of the work process, but also to simplify and improve the related tasks, such as the maintenance report creation. Working closely with the service staff and energy provider while creating and revising the user interface and interaction design as well as the application workflow ensures a usable template for the final software product. A prototype is currently being tested by Uptime Engineering and the energy provider to verify the feasibility of the approach, while Uptime Engineering works on integrating a user interface to the diagnosis engine within their condition monitoring system.

Nevertheless open issues in the fault identification of wind turbines still remain. Even after performing a repair, a failure might prevail in the system and its manifestations will be visible for a certain period of time. In this case, the diagnosis must be repeated and the original failure report should be re-opened. In addition, the automatic acquisition of additional expert knowledge is challenging. Updating the failure assessment and thus the abductive model cannot solely be performed by the user due to quality concerns of the inserted data in regard to its technical soundness and the knowledge engineering capabilities of the service staff. Nevertheless, the question remains as to whether there is a feasible way to have the maintenance personnel make additions to the model. Further refinements to the diagnosis results based on knowledge about damage accumulation and load history could improve the fault identification process. This type of information can be exploited to determine the remaining life time of components which subsequently can be used to derive

and update fault likelihoods. This realization is left to our industrial partner. Other common considerations concern maintenance decisions based on the expected cost accounting for all expenditures associated with the diagnostic task, such as probing, repair, or replacement activities. In the future other application domains, such as truck fleets, should clarify whether abductive reasoning is a suitable diagnosis approach for industrial domains in general.

Part IV

Conclusion and Future Work

Summary and Conclusion

10

> *A thinker sees his own actions as experiments and questions–as attempts to find out something. Success and failure are for him answers above all.*
>
> — **Friedrich Nietzsche**
> "*The Gay Science*". 1882.

The goal of this thesis has been to *develop a framework for facilitating the application of abductive model-based diagnosis in industrial practice* (**RO**), due to the fact that a noteworthy dissemination of practical model-based diagnosis software of technical systems is still missing. This work has been heavily influenced by the AMOR project in an undertaking to bridge the gap between theory and practice. In particular, we have worked on achieving our research objective from both a scientific and from an application-oriented perspective. On the theoretical side, we define a process for abductive model-based diagnosis in real-world operations and analyze algorithms for efficient generation of solutions in propositional logic. The main practical contribution of this thesis is a case study that connects the theoretical process to fault identification of industrial wind power plants. We conclude this work by summing up the efforts made towards our objectives and the findings uncovered on the way.

RO$_1$ *Develop a process for abductive model-based diagnosis in practice that (1) facilitates model generation, (2) can be easily integrated into current work processes, (3) allows for efficient abduction procedures, and (4) incorporates techniques to improve upon the initial diagnosis results.*

Chapter 3 contributes to achieving this first research objective by proposing a generally applicable process for integrating abductive model-based diagnosis tools in industrial domains. (1) Computation of root causes relies on the presence of a suitable system description. From a practical point of view, however, due to the unavailability of tools that can be used by people who are not experts in logical-based modeling, construction of an appropriate formalization of diagnostic expertise remains a hindering factor in the adoption of model-based diagnosis in real-world fault identification. As in the abductive variation the model characterizes the relationship between faults and their manifestations; the proposed general process takes advantage of failure assessments commonly used in practice for computerized knowledge base development. In our analysis, we have discussed FMEA, fault trees, and later in Chapter 8 we consider another failure record based on PoF. Automating this task should greatly facilitate the introduction of this diagnostic technique in industrial applications.

(2) We argue that the framework can be straightforwardly incorporated, as the process clearly splits the overall task into three parts: model generation, fault detection, and fault identification. The modeling process is executed in advance and requires that the failure assessment be available in a structured form, ideally as in FMEA in a tabulated manner. In our methodology, fault detection is seen as an external black box that triggers the diagnosis and provides a set of observations, while the fault identification segment is concerned with pinpointing the failure location. Of course, the communication between the parts is essential in regard to defining a common language respecting syntactic objectives. For instance, the

abductive engine relies on the detected anomalies and offline created model; thus the logical formalism used has to be appropriate for the diagnosis procedure.

(3) The used diagnostic reasoning engine is restricted only by the characteristic of the knowledge base and semantics of the observation. In the examined assessments, a simple Horn clause abduction procedure is sufficient to extract the required diagnoses. Here, the ATMS presents itself as a suitable tool, as it can be easily utilized not only for computing solutions but also as a means to determine the ideal measurement points to improve diagnosis results. Yet, any diagnostic engine capable of handling the underlying model formalism may be exploited as a simple back box to derive explanations. In Chapters 4 and 5, we evaluate different abduction procedures in regard to their efficiency.

(4) Besides automating the generation of system descriptions, the process incorporates an entropy-based method for probe selection to discriminate explanations and a probability-based solution ranking. As we have discussed, depending on the underlying assessment additional information, such as failure severity, may lend itself to be used for prioritizing diagnoses. Of course, in this regard a holistic approach considering probabilities, costs of maintenance activities, revenue loss of idle time or decreased system power, severities etc. would be beneficial in a practical application.

While Chapter 3 does contribute to the research objective, there are certain aspects that have been neglected within our analysis. For instance, the approach can only operate well under the assumption that all failures are in fact contained in the examination. This, even though commonly accepted in model-based diagnosis, may be a strong prerequisite from a practical perspective. To some extend, the OSFDP should supply indications in case certain information is missing from the analysis; yet, it cannot entirely overcome the model completeness assumption. Additionally, we have so far chosen only static failure records for compiling our system model. Investigating other formalisms, such as Modelica simulations [Pei+16], that are commonly used may address other application domains or be useful in order to retrieve additional knowledge absent from the failure assessments. Moreover, our process does not account for time or uncertainty in any way, i.e., the approach is affected by delayed observations, measurement inaccuracies, and sensor errors. Although the modeling proposed in Chapter 8 avoids some of these issues, it is not a comprehensive solution. While for our practical case study our industrial partner handles this concern in their failure detection procedure, considering it within the diagnosis portion would definitely be of interest in future work.

RO$_2$ *Explore procedures that enable an efficient computation of explanations for abduction problems obtained in a diagnosis scenario.*

Chapter 4 and 5 investigate the efficiency of abduction approaches. Based on the model structure and complexity results of Chapter 3, Chapter 4 aims at revealing efficient diagnosis procedures for system descriptions stemming from FMEAs. The model compiled on the basis of an FMEA contains bijunctive definite Horn clauses. Therefore abduction is tractable and the problems are equivalent to the bipartite diagnosis problems that are solved in the simple set-covering theory. Hence, we exploit the correspondence between set covers and hitting sets to formulate the solutions to a *PHCAP* as a minimal hitting set. Subsequently, we discuss several algorithms for deriving minimal hitting set and compare their efficiency on various diagnosis scenarios stemming from practical FMEAs and artificially constructed instances. In our first experimental setup, we could clearly identify that the Boolean approach and its strategy of splitting the search space are superior to the other methods. Considering again the diagnostic process of Chapter 3, we could recognize that while for certain application domains the technique would be ideal, unfortunately, the Boolean approach is not able of handling additional observations as often required in diagnosis. Hence, we exchanged in a

second experiment the Boolean algorithm with Berge's algorithm because it is capable of extracting hitting sets on the fly. While there was not a clear contender for the most efficient algorithm within this portfolio of approaches, we could identify that Berge and BHS-Tree observe very similar runtimes, while HS-DAG can outperform the two methods on other diagnosis problems. This observation is essential in motivating the work in Chapter 6.

Besides the hitting set formulation of abduction, we examine conflict-driven methods for computing explanations. By rewriting the diagnosis problem in an entailment preserving way, explanations are equivalent to refutations. Inspired by the advances in SAT solving, we utilize the hitting set duality between MUSes, i.e., conflicts, and MCSes to propose a direct and an indirect SAT-based abductive diagnosis procedure. We compare these two approaches with the ATMS and the conflict-driven search of HS-DAG as it has been traditionally used in consistency-based diagnosis. Unsurprising, we could observe that utilizing tools not tailored towards assumption-based reasoning and Horn clause theories, such as MUS and MCS enumeration techniques, cannot compete with the specialized reasoners. Although the ATMS was not superior for bipartite abduction problems, it is preferable on the FMEA samples in comparison to conflict directed procedures.

Chapter 5 aims at providing guidance on choosing an algorithm or tool when confronted with the issue of computing explanations in propositional logic-based abduction. Our focus lies on a Horn representation as it provides a suitable language to describe most diagnostic scenarios. In this context, we discuss a set of abduction procedures via two contrasting problem formulations: direct proof methods and conflict-driven techniques. To reveal runtime performance trends, we conducted a case study, in which we compared publicly available general-purpose tools, established Horn reasoning engines as well as new variations of known methods as a means for abduction. We could not determine that either the direct or the conflict-driven approaches are in general dominant on the samples. Yet, on all benchmarks, we could again observe that the ATMS and HS-DAG are competitive, where this version of HS-DAG is improved by ensuring that conflicts are minimized right away before continuing to construct the DAG. While the general off-the-shelf tools, i.e., consequence finding procedure SOLAR and ASP solver clingo, were not contenders for computing the diagnoses the fastest, both managed to solve the majority of the problem instances given the time constraints applied to the experiments.

Concluding both chapters, we can state that for the simple bipartite problems using an efficient hitting set approach such as Berge's algorithm or HS-DAG is preferable over direct techniques such as the ATMS and conflict-driven approaches. We can deduce this as we have carried out similar experiments based on FMEA and artificial samples on the hitting set and the conflict-directed techniques. For the more expressive Horn clauses, we deduce that the conflict-driven usage of HS-DAG and the ATMS have proven advantageous in our evaluations, and thus aided us in achieving **RO$_2$**.

For our evaluations in Chapter 4 and 5 there are obviously always more options of algorithms and tools to consider. While our experiments indicate trends among the selected set of approaches, a more in-depth analysis of the causing algorithm properties would be desirable. In addition, our experiments—as every experimental evaluation—are limited in the number, characteristics, and extensiveness of the utilized benchmarks. While there are the ISCAS circuits[1] for consistency-based diagnosis as a well-known and generally accepted test bed, within the abductive model-based diagnosis community there does not exist such a benchmark. Another apparent limitation of our empirical work is the influence the programming language may have on the efficiency of the different abduction procedures,

[1] We have conducted experiments, where we attempted at abductively solving the ISCAS'85 circuits using the ATMS. However, we ran into label explosions and thus unacceptable runtimes early on in the experiments and hence aborted our initiative in this regard.

especially in regard to evaluations of hitting set algorithms in Chapter 4. Here we refer the interested reader to Pill, Quaritsch, and Wotawa [Pil+11], who explicitly compared Java and Python implementations of hitting set procedures.

RO$_3$ *Optimize the reasoning approaches to be suitable for practical applications in regard to efficiency.*

In Chapter 6, we investigate algorithm selection as a method to improve the efficiency of abductive reasoning and thus work on **RO$_2$** and **RO$_3$**. First, we examine the structural properties inherent to bijunctive definite Horn theories stemming from FMEAs and propositional Horn clause models that allow evaluating the difficulty of a problem instance. Based on the features extracted, we construct a meta-approach (MetAB) taking advantage of a machine learning classifier to predict the abductive reasoning technique yielding the "best" performance on a specific diagnosis scenario. MetAB's overall runtime is determined by (1) the computation of the online metrics, (2) the time it takes to create the feature vector, supply it to the classifier, and predict the "best" algorithm, and (3) the diagnosis time of the suggested abduction procedure. To assess whether the proposed attributes are in fact sufficient for forecasting the appropriate abduction procedure and to evaluate the efficiency of our algorithm selection in comparison to traditional abductive reasoning approaches, we carried out empirical experiments. First, we rely on the same performance data as in Chapter 4. Based on the simple bipartite diagnosis problems, we could not determine a benefit of our meta-approach. The reason lies in the simplicity of these models, where the computation of the online features requires around the same time as the diagnosis itself. Hence, MetAB introduces on these instances a computational overhead. Second, based on the performance data of Chapter 5 utilizing Horn theory models, the evaluation indicates that the trained model is able of predicting the most efficient algorithm with satisfactory accuracy and further, we can show that the meta-approach is capable of outperforming each single abductive reasoning method investigated at least for the artificial Horn abduction problems.

The great benefit of applying algorithm selection to model-based diagnosis is that a majority of features can be extracted based on the system description that has to be known in advance. Thus, the amount of features computed online is limited. This characteristic, however, can only be advantageous in case the extraction of the online metrics is rather brief in comparison to computing the actual diagnosis. To summarize, on simple diagnosis problems there is no advantage to be gained from using algorithm selection, but on more expressive samples the meta-approach becomes valuable. Missing from our work is an evaluation with more extensive examples on which we assume our meta-approach would perform particularly well. Further, the attributes we have applied are rather straightforward. There are certainly possibilities to extract more particular features from the models and in addition improve the attribute selection process to identify metrics (and their combinations) that increase the prediction accuracy more drastically.

RO$_4$ *Analyze current wind turbine diagnosis techniques and identify available data and tools.*

The number and complexity of industrial wind turbine installations have increased significantly over the last decades. As maintenance costs are high and down-times lead to substantial revenue loss, increasing the reliability and optimizing the maintenance process are crucial tasks from an industrial perspective. Analyzing the related research in Chapter 7 in regard to wind turbine fault identification, we could determine that many of the proposed diagnosis systems merely focus on parts of the turbine or locate only a portion of the faults. Therefore, we propose a model-based approach depending on automatically retrieved health

variables from the turbine itself and on extensive expert knowledge on specific component-oriented failure modes as well as their effects on measurable signals, i.e., FMA. As the expert assessment provides causal links between faults and their manifestations, we can utilize this knowledge for abductive reasoning.

A detailed description of how to handle sensor inaccuracies or the way in which aggravating boundary conditions and damage promoting operation modes will be incorporated in detail is missing in this thesis. To some extent, fallacies due to measurement errors are avoided by our knowledge compilation to a Horn model as discussed in Chapter 8. In addition, aggravating boundary conditions and damage promoting operation modes can be incorporated as additional information sources to define a ranking of diagnoses; yet, the details of the practical solution are left to our industrial partner.

RO$_5$ *Identify modifications to the theoretical process necessary to match the information available and the domain requirements.*

A common constraint of diagnosis engines is to restrict the model to comprise Horn clauses. As we have seen in Chapter 3, we compile FMEAs and fault trees into abduction knowledge bases consisting of sentences of this subset of logics. There are, however, practical applications where this limitation is problematic as the failure behavior of the system would require a more expressive representation. In our AMOR project, the structure and information of the underlying failure assessment changed thereby making it crucial to adapt the methodology for compiling the system description. In Chapter 8, we describe a modeling method for abductive diagnosis based on an extension of Horn logic, which allows expressing conjunctions and disjunctions of effects. In Section 8.4, we show that structured assessments based on the PoF approach our industrial partner is using may contain this type of knowledge. Hence, we present a mapping from this representation to a Horn theory. Besides the fact that such a model can be applied to a wide range of diagnosis algorithms, the essential contribution of this chapter is that diagnosis based on the models converted to Horn provides intuitive explanations similar to human reasoning. This is especially favorable in the context of practical fault identification, where acceptance of knowledge-based systems largely benefits from solutions being comprehensible to the user.

We presented initial results comparing diagnosis on the ATMS and an ASP solver in regard to the Horn transformation. To obtain our intuitive explanations, we needed to adapt the models for the ASP framework. Ideally, we would modify the encoding used for abduction, to allow deriving these explanations right away from the compiled model we also use for the ATMS. In addition, our work lacks a semantic formalization of these intuitive diagnoses.

RO$_6$ *Show how to integrate an automated modeling and diagnosis approach into current work processes.*

In Chapter 9, we have developed a concept and interface design for an abductive model-based diagnosis application based on the process of Chapter 3. Currently, this system is being integrated into our industrial partner's existing condition monitoring software for industrial wind turbines. This chapter emphasizes the role of essential technology acceptance factors, i.e., usefulness and usability, within the context of model-based diagnosis. While model-based diagnosis research is mostly concerned with developing new techniques or improving upon well-known approaches, the intersection of model-based diagnosis and user experience and design is often overlooked. The outlined fault identification tool should enhance the performance of the maintenance personnel while respecting their current work processes, take into account their particular needs, and be easy to use under the given work conditions. By employing an iterative design process, continuous feedback in regard to the users' work goals, tasks, and patterns was included, while also considering other

stakeholders' requirements. The result is a workflow and interface layout proposal to be implemented in the final software product. Further, we have discussed the current stage of the application's integration into the wind turbine diagnosis process, i.e., while the back-end has been developed the user interface is still under construction hindering a broader evaluation of the software. Yet, informal analyses have already been conducted and could confirm that the diagnosis results and the general operation of the application are fulfilling the requirements.

An essential aspect that we are currently handling in a simple manner is the knowledge acquisition problem that all knowledge-based systems face. In particular, the human operator may observe certain circumstances that have not been anticipated within the failure assessment and thus are not mapped onto the abductive knowledge base. However, these additional failure modes or observations to a fault should be integrable into the model. Here, the question is whether the service technicians are capable and should be able to adapt the underlying system description or whether this would always have to be done by an expert in the field of reliability analysis. In case the users do possess the required competence and experience to augment the model there is still the need to develop a structured way of adding this type of information. Another interface matter that may improve the application's usefulnesses, is an explanation component that illustrates how the system has come up with the solution in order to facilitate acceptance by the service staff.

11 Future Work

> *... no sooner is a discovery or invention made, than it is already improved upon and surpassed by competing efforts.*
>
> **— Prince Albert of Saxe-Coburg and Gotha**
> *Inaugural Address of the "Great Exhibition of the Works of Industry of All Nations"*. 1851.

Besides attending to the limitations of our research, which we have discussed throughout this work and in the previous section, there are a number of interesting ideas that could not be addressed in the scope of this thesis. From the analyses of the abduction methods, we believe there is potential to further improve upon the methods. In particular, it seems that the way HS-DAG is restricting the search space is advantageous. It would be interesting to examine whether other conflict extraction methods, such as MergeXplain that generate not only a single conflict but possibly several, would be beneficial. In regard to the exploration of the power lattice, we can improve the method we have sketched out in Chapter 5. As we seek to only extract MUS and are not necessarily interested in deriving MSS, blocking with each found unsatisfiable seed up and down at the same time reduces the search space. This, also diminishes the number of seeds necessary to explore the entire lattice. As the computation of the maximal model for the seed has been contributing largely to the overall runtime, this simple adjustment may allow for a more efficient extraction of conflicts within this approach.

Further, we have applied only a simple version of the portfolio approach; there are various advanced methods that could be beneficial, such as switching between computation methods. This would be particularly interesting by exploiting whether certain approaches favor a kind of explanation and would produce those early on in the computation. By switching between those methods then and adding blocking clauses to ensure only not previously derived diagnoses are obtained, we may be able to improve the computation even further. These switching procedures may increase to a certain extent the computation time, and thus would probably be better suited on larger models.

Of course, while our main goal is to allow for general applications to utilize abductive model-based diagnosis, we have conducted a single case study in the wind turbine domain. Certain aspects of this field are in our favor, e.g., diagnoses need only be computed in the span of several hours, a fault identification procedure is already implemented, failure assessments are available etc. In other industries the requirements and environments may change; thus, a natural extension of our work would attempt to apply the process to another industrial field to verify whether it is applicable. This ties in with the idea of the automatic model creation from other expert knowledge available. Besides reliability analysis in the wind turbine domain, our industrial partner further develops vehicle fleet monitoring software, which may be another application area for our diagnostic methods.

Another aspect that has come up while working on the application concept with the service technicians is that in their experience there are certain turbines that are more prone to error despite being the same build as other turbines within the fleet. Given their work on the wind power plants, these particularities are known to them. Currently, however, there is no way for us to include or take advantage of such knowledge. A remedy may be a combination of

case-based reasoning and model-based diagnosis to augment the knowledge on failures and their effects with information from previous maintenance cases of individual turbines.

Bibliography

[Abd+03] Ashraf M Abdelbar, Emad AM Andrews, and Donald C Wunsch II. „Abductive reasoning with recurrent neural networks“. In: *Neural Networks* 16.5 (2003), pp. 665–673.

[AG09] Rui Abreu and Arjan J. C. van Gemund. „A Low-Cost Approximate Minimal Hitting Set Algorithm and its Application to Model-Based Diagnosis“. In: *Eighth Symposium on Abstraction, Reformulation, and Approximation, SARA 2009, Lake Arrowhead, California, USA, 8-10 August 2009*. 2009.

[Alt+89] Klaus-Dieter Althoff, S Kockskämper, R Traphöner, and W Wernicke. „Knowledge acquisition in the domain of CNC machining centers: the MOLTKE approach“. In: *Third European Workshop on Knowledge-Based Systems*. 1989.

[Ang+05] Christian Anger, Kathrin Konczak, Thomas Linke, and Torsten Schaub. „A Glimpse of Answer Set Programming“. In: *Künstliche Intelligenz* 19.1 (2005), pp. 12–17.

[Ari+15] M Fareed Arif, Carlos Mencía, and Joao Marques-Silva. „Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing“. In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2015, pp. 324–342.

[Ass14] European Wind Energy Association. *Wind in power - 2013 European statistics*. 2014. URL: `http://www.ewea.org/fileadmin/files/library/publications/statistics/EWEA_Annual_Statistics_2013.pdf`.

[Aus+80] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. „Structure preserving reductions among convex optimization problems“. In: *Journal of Computer and System Sciences* 21.1 (1980), pp. 136–153.

[Aye+98] Béchir el Ayeb, Shengrui Wang, and Jifeng Ge. „A unified model for abduction-based reasoning“. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 28.4 (1998), pp. 408–425.

[Ber89] Claude Berge. *Hypergraphs, Volume 45 of North-Holland Mathematical Library*. 1989.

[Bes+08] Ph Besnard, M-O Cordier, and Yves Moinard. „Ontology-based inference for causal explanation“. In: *Integrated Computer-Aided Engineering* 15.4 (2008), pp. 351–367.

[Bie09] Meghyn Bienvenu. „Prime implicates and prime implicants: From propositional to modal logic“. In: *Journal of Artificial Intelligence Research* (2009).

[Bit08] Guilherme Bittencourt. „Combining syntax and semantics through prime form representation“. In: *Journal of Logic and Computation* 18.1 (2008), pp. 13–33.

[Bot+12] Pantelis N Botsaris, EI Konstantinidis, and D Pitsa. „Systemic assessment and analysis of factors affect the reliability of a wind turbine“. In: *Journal of Applied Engineering Science (Istrazivanja i projektovanja za privredu)* 10.2 (2012).

[Bru+93] Simona Brugnoni, Guido Bruno, Roberto Manione, et al. „An expert system for real time fault diagnosis of the Italian telecommunications network“. In: *Proceedings of the IFIP TC6/WG6. 6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*. North-Holland Publishing Co. 1993, pp. 617–628.

[Bru+98] Vittorio Brusoni, Luca Console, Paolo Terenziani, and Daniele Theseider Dupré. „A spectrum of definitions for temporal model-based diagnosis“. In: *Artificial Intelligence* 102.1 (1998), pp. 39–79.

[BS02] Joachim Baumeister and Dietmar Seipel. „Diagnostic reasoning with multilevel set-covering models“. In: *Proceedings of the 13th International Workshop on Principles of Diagnosis (DX-02)*. 2002, pp. 1–7.

[BS84] Bruce G Buchanan and Edward H Shortliffe. „Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project“. In: *The Addison-Wesley Series in Artificial Intelligence* (1984).

[Byl+91] Tom Bylander, Dean Allemang, Michael C Tanner, and John R Josephson. „The computational complexity of abduction“. In: *Artificial Intelligence* 49.1-3 (1991), pp. 25–60.

[Car14] Carl S Carlson. „Understanding and Applying the Fundamentals of FMEAs“. In: *Proceedings of the 2014 Annual Reliability and Maintainability Symposium*. 2014.

[Cat+10] Marcantonio Catelani, Lorenzo Ciani, and Valentina Luongo. „The FMEDA approach to improve the safety assessment according to the IEC61508“. In: *Microelectronics Reliability* 50.9 (2010), pp. 1230–1235.

[CD99] Luca Console and Oskar Dressier. „Model-based diagnosis in the real world: Lessons learned and challenges remaining“. In: *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc. 1999, pp. 1393–1400.

[Che+12] Bindi Chen, Peter Tavner, Yanhui Feng, William W Song, and Yingning Qiu. „Bayesian network for wind turbine fault diagnosis“. In: (2012).

[Con+91] Luca Console, Daniele Theseider Dupré, and Pietro Torasso. „On the relationship between abduction and deduction“. In: *Journal of Logic and Computation* 1.5 (1991), pp. 661–690.

[Con+96] Luca Console, Luigi Portinale, and D Theseider Dupré. „Using compiled knowledge to guide and focus abductive diagnosis“. In: *IEEE Transactions on Knowledge and Data Engineering* 8.5 (1996), pp. 690–706.

[Cor+04] M-O Cordier, Philippe Dague, François Lévy, et al. „Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.5 (2004), pp. 2163–2177.

[CP86] Philip T Cox and Tomasz Pietrzykowski. „Causes for events: their computation and applications“. In: *International Conference on Automated Deduction*. Springer. 1986, pp. 608–621.

[CR01] Roman Cunis and Peter Roman. „INDIA—Intelligente Diagnose in der industriellen Anwendung“. In: *Intelligent Diagnosis in Industrial Applications* (2001), pp. 1–9.

[CS94] Eugene Charniak and Solomon Eyal Shimony. „Cost-based abduction and MAP explanation“. In: *Artificial Intelligence* 66.2 (1994), pp. 345–374.

[CT06] Luca Console and Pietro Torasso. „Automated diagnosis“. In: *Intelligenza Artificiale* 3.1-2 (2006), pp. 42–48.

[CT91] Luca Console and Pietro Torasso. „A spectrum of logical definitions of model-based diagnosis“. In: *Computational Intelligence* 7.3 (1991), pp. 133–141.

[Dav89] Fred D Davis. „Perceived usefulness, perceived ease of use, and user acceptance of information technology“. In: *MIS quarterly* (1989), pp. 319–340.

[DH88] Randall Davis and Walter Hamscher. „Model-based reasoning: Troubleshooting“. In: *Exploring artificial intelligence* 8 (1988), pp. 297–346.

[DK02] Marc Denecker and Antonis Kakas. „Abduction in logic programming“. In: *Computational logic: Logic programming and beyond* (2002), pp. 99–134.

[DK86a] Johan De Kleer. „An assumption-based TMS“. In: *Artificial Intelligence* 28.2 (1986), pp. 127–162.

[DK86b] Johan De Kleer. „Extending the ATMS“. In: *Artificial Intelligence* 28.2 (1986), pp. 163–196.

[DK86c] Johan De Kleer. „Problem solving with the ATMS“. In: *Artificial Intelligence* 28.2 (1986), pp. 197–224.

[DKW87] Johan De Kleer and Brian C Williams. „Diagnosing multiple faults“. In: *Artificial Intelligence* 32.1 (1987), pp. 97–130.

[DM02] Adnan Darwiche and Pierre Marquis. „A knowledge compilation map“. In: *Journal of Artificial Intelligence Research* 17.1 (2002), pp. 229–264.

[DP91] Jon Doyle and Ramesh S Patil. „Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services“. In: *Artificial Intelligence* 48.3 (1991), pp. 261–297.

[DP99] Oskar Dressler and Frank Puppe. „Knowledge-Based Diagnosis–Survey and Future Directions“. In: *German Conference on Knowledge-Based Systems*. Springer. 1999, pp. 24–46.

[Dub+91] Didier Dubois, Jéřome Lang, and Henri Prade. „A possibilistic assumption-based truth maintenance system with uncertain justifications, and its application to belief revision“. In: *Truth maintenance systems* (1991), pp. 87–106.

[Ech+07] Erika Echavarria, Tetsuo Tomiyama, and Gerard J.W. van Bussel. „Fault Diagnosis approach based on a model-based reasoner and a functional designer for a wind turbine. An approach towards self-maintenance“. In: *Journal of Physics: Conference Series* 75 (July 2007), p. 012078. ISSN: 1742-6596. URL: `http://stacks.iop.org/1742-6596/75/i=1/a=012078?key=crossref.dce18ef408282263748fb70ab56f3a46`.

[EG95] Thomas Eiter and Georg Gottlob. „The complexity of logic-based abduction“. In: *Journal of the ACM (JACM)* 42.1 (1995), pp. 3–42.

[Egl+00] Uwe Egly, Thomas Eiter, Hans Tompits, and Stefan Woltran. „Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas“. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press. 2000, pp. 417–422.

[Eit+09] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. „Answer set programming: A primer“. In: *Reasoning Web. Semantic Technologies for Information Systems*. Springer, 2009, pp. 40–110.

[Eit+98] Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarcello. „The KR system dlv: Progress report, comparisons and benchmarks“. In: *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc. 1998, pp. 406–417.

[Eit+99] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. „The diagnosis frontend of the dlv system“. In: *AI Communications* 12.1-2 (1999), pp. 99–111.

[EK89] Kave Eshghi and Robert A Kowalski. „Abduction Compared with Negation by Failure“. In: *Proceedings 6th International Conference on Logic Programming*. Vol. 89. 1989, pp. 234–255.

[EM03] Thomas Eiter and Kazuhisa Makino. „Generating All Abductive Explanations for Queries on Propositional Horn Theories“. In: *International Workshop on Computer Science Logic*. Springer. 2003, pp. 197–211.

[ES03] Niklas Eén and Niklas Sörensson. „An extensible SAT-solver“. In: *International Conferemce on theory and applications of SAT*. Springer. 2003, pp. 502–518.

[FB+97] Dieter Fensel, Richard Benjamins, et al. *Assumptions in model-based diagnosis*. AIFB, Univ, 1997.

[Fel+04] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumptner. „Consistency-based diagnosis of configuration knowledge bases“. In: *Artificial Intelligence* 152.2 (2004), pp. 213–234.

[Fel+10a] Alexander Feldman, Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, and David Garcia. „Empirical Evaluation of Diagnostic Algorithm Performance Using a Generic Framework“. In: *International Journal of Prognostics and Health Management Volume 1* (2010), p. 24.

[Fel+10b] Alexander Feldman, Gregory Provan, Johan de Kleer, Stephan Robert, and Arjan van Gemund. „Solving model-based diagnosis problems with max-sat solvers and vice versa“. In: *Proceedings of the 21th International Workshop on Principles of Diagnosis (DX-10)*. 2010, pp. 185–192.

[Fel+13] Alexander Felfernig, Monika Schubert, and Stefan Reiterer. „Personalized diagnosis for over-constrained problems“. In: *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press. 2013, pp. 1990–1996.

[FG85] JJ Finger and Michael R Genesereth. „RESIDUE: a deductive approach to design synthesis“. In: (1985).

[FK88] Kenneth D Forbus and Johan de Kleer. „Focusing the ATMS“. In: *Proceedings of the Seventh AAAI Conference on Artificial Intelligence*. AAAI Press. 1988, pp. 193–198.

[Fle+01] Gerhard Fleischanderl, Thomas Havelka, Herwig Schreiner, Markus Stumptner, and Franz Wotawa. „DiKe-A Model-Based Diagnosis Kernel and Its Application“. In: *Proceedings of the Joint German/Austrian Conference on AI: Advances in Artificial Intelligence*. Springer-Verlag. 2001, pp. 440–454.

[FN97] Peter Fröhlich and Wolfgang Nejdl. „A static model-based engine for model-based reasoning“. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. 1997, pp. 466–473.

[For+13] Nathan Fortier, John Sheppard, and Karthik Ganesan Pillai. „Bayesian abductive inference using overlapping swarm intelligence“. In: *Swarm Intelligence (SIS), 2013 IEEE Symposium on*. IEEE. 2013, pp. 263–270.

[Fri+90a] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. „Hypothesis classification, abductive diagnosis and therapy“. In: *Expert Systems in Engineering Principles and Applications* (1990), pp. 69–78.

[Fri+90b] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. „Physical impossibility instead of fault models“. In: *Proceedings of the eighth National conference on Artificial intelligence-Volume 1*. AAAI Press. 1990, pp. 331–336.

[Fri+99] Gerhard Friedrich, Markus Stumptner, and Franz Wotawa. „Model-based diagnosis of hardware designs“. In: *Artificial Intelligence* 111.1 (1999), pp. 3–39.

[FS10] Alexander Felfernig and Monika Schubert. „FastDiag: A diagnosis algorithm for inconsistent constraint sets“. In: *Proceedings of the 21st International Workshop on the Principles of Diagnosis (DX 2010)*. 2010, pp. 1–8.

[Gan+05] Sathyanarayan Ganesan, Valerie Eveloy, Diganta Das, and M Pecht. „Identification and utilization of failure mechanisms to enhance FMEA and FMECA“. In: *Proceedings of the IEEE workshop on accelerated stress testing & reliability (ASTR), Austin, Texas*. Vol. 3. 4. 2005, p. 5.

[Geb+14] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. „Clingo = ASP + Control: Preliminary Report“. In: *In Technical Communications of the 30th International Conference on Logic Programming*. Ed. by M. Leuschel and T. Schrijvers. 2014, pp. 1–13.

[GH07] Haipeng Guo and William H Hsu. „A machine learning approach to algorithm selection for NP-hard optimization problems: a case study on the MPE problem“. In: *Annals of Operations Research* 156.1 (2007), pp. 61–82.

[Got+90] Georg Gottlob, Thomas Frühwirth, and Werner Horn. *Expertensysteme*. Springer Vienna, 1990.

[Gra+11] Christopher Gray, Franz Langmayr, Nikolaus Haselgruber, and Simon J Watson. „A practical approach to the use of scada data for optimized wind turbine condition based maintenance“. In: *EWEA Offshore Wind Amsterdam* (2011).

[Gra+15] Christopher S Gray, Roxane Koitz, Siegfried Psutka, and Franz Wotawa. „An Abductive Diagnosis and Modeling Concept for Wind Power Plants“. In: *IFAC-PapersOnLine* 48.21 (2015), pp. 404–409.

[Gre+89] Russell Greiner, Barbara A Smith, and Ralph W Wilkerson. „A correction to the algorithm in Reiter's theory of diagnosis“. In: *Artificial Intelligence* 41.1 (1989), pp. 79–88.

[Gua17] The Guardian. *Oil jumps after Forties pipeline crack discovered*. 2017. URL: `https://www.theguardian.com/business/live/2017/dec/12/oil-brent-crude-forties-oil-pipeline-uk-inflation-business`.

[Gur17] Dmitriy Gurkovskiy. *Oil Calmed Down After Forties Pipeline System Failure*. 2017. URL: `https://seekingalpha.com/article/4132810-oil-calmed-forties-pipeline-system-failure`.

[GW10] Christopher S Gray and Simon J Watson. „Physics of failure approach to wind turbine condition based maintenance“. In: *Wind Energy* 13.5 (2010), pp. 395–405.

[Hal+09] Mark Hall, Eibe Frank, Geoffrey Holmes, et al. „The WEKA data mining software: an update“. In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18.

[Ham+09] Z Hameed, YS Hong, YM Cho, SH Ahn, and CK Song. „Condition monitoring and fault detection of wind turbines and related algorithms: A review“. In: *Renewable and Sustainable energy reviews* 13.1 (2009), pp. 1–39.

[Hob+93] Jerry R Hobbs, Mark E Stickel, Douglas E Appelt, and Paul Martin. „Interpretation as abduction“. In: *Artificial Intelligence* 63.1-2 (1993), pp. 69–142.

[Hut+06] Frank Hutter, Youssef Hamadi, Holger Hoos, and Kevin Leyton-Brown. „Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms“. In: *Principles and Practice of Constraint Programming-CP 2006* (2006), pp. 213–228.

[HW98] P G Hawkins and David J Woollons. „Failure modes and effects analysis of complex engineering systems using functional models“. In: *Artificial Intelligence in Engineering* 12.4 (1998), pp. 375–397.

[Ign+16] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. „Propositional Abduction with Implicit Hitting Sets“. In: *arXiv preprint arXiv:1604.08229* (2016).

[Ino02] Katsumi Inoue. „Automated abduction“. In: *Computational Logic: Logic Programming and Beyond* (2002), pp. 35–72.

[Ino91] Katsumi Inoue. „An abductive procedure for the CMS/ATMS“. In: *Truth Maintenance Systems* (1991), pp. 34–53.

[Ino92] Katsumi Inoue. „Linear resolution for consequence finding“. In: *Artificial Intelligence* 56.2 (1992), pp. 301–353.

[Jan+15] Dietmar Jannach, Thomas Schmitz, and Kostyantyn Shchekotykhin. „Parallelized hitting set computation for model-based diagnosis“. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press. 2015, pp. 1503–1510.

[Jun04] Ulrich Junker. „QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems“. In: *Proceedings of the 19th national conference on Artifical intelligence*. AAAI Press. 2004, pp. 167–172.

[Kak+01] Antonis C Kakas, Bert Van Nuffelen, and Marc Denecker. „A-system: Problem solving through abduction“. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. 2001, pp. 591–596.

[Kak+92] Antonis C. Kakas, Robert A. Kowalski, and Francesca Toni. „Abductive logic programming“. In: *Journal of logic and computation* 2.6 (1992), pp. 719–770.

[Kav+12] Soila P Kavulya, Kaustubh Joshi, Felicita Di Giandomenico, and Priya Narasimhan. „Failure diagnosis of complex systems“. In: *Resilience assessment and evaluation of computing systems*. Springer, 2012, pp. 239–261.

[KHW18] Roxane Koitz-Hristov and Franz Wotawa. „Applying Algorithm Selection to Abductive Diagnostic Reasoning“. In: *Applied Intelligence* (2018). ISSN: 1573-7497. URL: `https://doi.org/10.1007/s10489-018-1171-9`.

[Kle92] Johan de Kleer. „An improved incremental algorithm for generating prime implicates“. In: *Proceedings of the tenth national conference on Artificial intelligence*. AAAI Press. 1992, pp. 780–785.

[Koi+17] Roxane Koitz, Johannes Lüftenegger, and Franz Wotawa. „Model-Based Diagnosis in Practice: Interaction Design of an Integrated Diagnosis Application for Industrial Wind Turbines.“ In: *Proceedings of the 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*. 2017, pp. 440–445.

[Koi+18] Roxane Koitz, Franz Wotawa, Johannes Lüftenegger, Chrsitopher S. Gray, and Franz Langmayr. „Wind Turbine Fault Localization: A Practical Application of Model-Based Diagnosis“. In: *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*. Cham: Springer International Publishing, 2018, pp. 17–43. ISBN: 978-3-319-74962-4. URL: `https://doi.org/10.1007/978-3-319-74962-4_2`.

[Kot14] Lars Kotthoff. „Algorithm selection for combinatorial search problems: A survey“. In: *AI Magazine* 35.3 (2014), pp. 48–60.

[KT90] Alex Kean and George Tsiknis. „An incremental method for generating prime implicants/implicates“. In: *Journal of Symbolic Computation* 9.2 (1990), pp. 185–206.

[Kun+11] Fabien Kuntz, Stéphanie Gaudan, Christian Sannino, et al. „Model-based diagnosis for avionics systems using minimal cuts“. In: *Proceedings of the 22nd International Workshop on Principles of Diagnosis*. 2011, pp. 138–145.

[KW15a] Roxane Koitz and Franz Wotawa. „Finding Explanations: An Empirical Evaluation of Abductive Diagnosis Algorithms“. In: *Proceedings of the 2015 International Workshop on Defeasible and Ampliative Reasoning*. CEUR-WS. org. 2015, pp. 36–42.

[KW15b] Roxane Koitz and Franz Wotawa. „From Theory to Practice: Model-Based Diagnosis in Industrial Applications“. In: *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society*. 2015, pp. 197–205.

[KW15c] Roxane Koitz and Franz Wotawa. „On the Computational Feasibility of Abductive Diagnosis for Practical Applications“. In: *IFAC-PapersOnLine* 48.21 (2015). 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, pp. 410–415.

[KW15d] Roxane Koitz and Franz Wotawa. „SAT-Based Abductive Diagnosis“. In: *26th International Workshop on Principles of Diagnosis*. 2015, pp. 1–9.

[KW16a] Roxane Koitz and Franz Wotawa. „Improving Abductive Diagnosis Through Structural Features: A Meta-Approach“. In: *Proceedings of the 2016 International Workshop on Defeasible and Ampliative Reasoning*. 2016, pp. 1–9.

[KW16b] Roxane Koitz and Franz Wotawa. „Integration of Failure Assessments into The Diagnostic Process“. In: *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society*. 2016, pp. 117–128.

[KW16c] Roxane Koitz and Franz Wotawa. „On Structural Properties to Improve FMEA-Based Abductive Diagnosis“. In: *Proceedings of the Workshop on Knowledge-based Techniques for Problem Solving and Reasoning*. 2016, pp. 1–7.

[LB+03] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. „A portfolio approach to algorithm select“. In: *Proceedings of the 18th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc. 2003, pp. 1542–1543.

[Lev89] Hector J Levesque. „A knowledge-level account of abduction“. In: *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc. 1989, pp. 1061–1067.

[Lif+16] Mark H Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. „Fast, flexible MUS enumeration“. In: *Constraints* 21.2 (2016), pp. 223–250.

[Lif02] Vladimir Lifschitz. „Answer set programming and plan generation“. In: *Artificial Intelligence* 138.1-2 (2002), pp. 39–54.

[LJ03] Li Lin and Yunfei Jiang. „The computation of hitting sets: review and new algorithms“. In: *Information Processing Letters* 86.4 (2003), pp. 177–184.

[LS08] Mark H Liffiton and Karem A Sakallah. „Algorithms for computing minimal unsatisfiable subsets of constraints“. In: *Journal of Automated Reasoning* 40.1 (2008), pp. 1–33.

[Lu+09] Bin Lu, Yaoyu Li, Xin Wu, and Zhongzhou Yang. „A review of recent advances in wind turbine condition monitoring and fault diagnosis“. In: *Power Electronics and Machines in Wind Applications, 2009. PEMWA 2009. IEEE*. IEEE. 2009, pp. 1–7.

[Lüf18] Johannes Lüftenegger. „Development and Evaluation of a User Interface Concept for an Industrial Wind Turbine Diagnosis Application“. Master's thesis. Graz University of Technology, 2018.

[Mal+14] Yuri Malitsky, Barry O'Sullivan, Alessandro Previti, and Joao Marques-Silva. „A portfolio approach to enumerating minimal correction subsets for satisfiability problems“. In: *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*. Springer. 2014, pp. 368–376.

[Mal13] Ammar Malik. „Analyzing and Extending an Infeasibility Analysis Algorithm“. In: *Honors Projects.* (2013).

[Man+97] Vasco M Manquinho, Paulo F Flores, Joao P Marques Silva, and Arlindo L Oliveira. „Prime implicant computation using satisfiability algorithms“. In: *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*. IEEE. 1997, pp. 232–239.

[Mar00] Pierre Marquis. „Consequence finding algorithms“. In: *Handbook of Defeasible Reasoning and Uncertainty Management Systems*. Springer, 2000, pp. 41–145.

[Mat+09] Andrew Matusiewicz, Neil V Murray, and Erik Rosenthal. „Prime implicate tries“. In: *Automated Reasoning with Analytic Tableaux and Related Methods*. Springer, 2009, pp. 250–264.

[McI94] Sheila A McIlraith. „Generating tests using abduction“. In: *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc. 1994, pp. 449–460.

[McI98] Sheila A McIlraith. „Logic-based abductive inference“. In: *Knowledge Systems Laboratory, Technical Report KSL-98-19* (1998).

[Met+14] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. „A novel sat-based approach to model based diagnosis“. In: *Journal of Artificial Intelligence Research* 51 (2014), pp. 377–411.

[Mil+00] Heiko Milde, Thomas Guckenbiehl, Andreas Malik, Bernd Neumann, and Peter Struss. „Integrating model-based diagnosis techniques into current work processes–three case studies from the INDIA project“. In: *AI Communications* 13.2 (2000), pp. 99–123.

[Mil+82] Randolph A Miller, Harry E Pople Jr, and Jack D Myers. „Internist-I, an experimental computer-based diagnostic consultant for general internal medicine“. In: *New England Journal of Medicine* 307.8 (1982), pp. 468–476.

[Min88] Michel Minoux. „LTUR: A simplified linear-time unit resolution algorithm for horn formulae and computer implementation“. In: *Information Processing Letters* 29.1 (1988), pp. 1–12.

[Moi10] Yves Moinard. „A Formalism for Causal Explanations with an Answer Set Programming Translation“. In: *KSEM*. Springer. 2010, pp. 585–590.

[MS+13] Joao Marques-Silva, Federico Heras, Mikolas Janota, Alessandro Previti, and Anton Belov. „On computing minimal correction subsets“. In: *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press. 2013, pp. 615–622.

[MS+15] Joao Marques-Silva, Mikoláš Janota, Alexey Ignatiev, and Antonio Morgado. „Efficient Model Based Diagnosis with Maximum Satisfiability“. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. Vol. 15. 2015, pp. 1966–1972.

[Már+12] Fausto Pedro García Márquez, Andrew Mark Tobias, Jesús María Pinar Pérez, and Mayorkinos Papaelias. „Condition monitoring of wind turbines: Techniques and methods“. In: *Renewable Energy* 46 (2012), pp. 169–178.

[Nab+10] Hidetomo Nabeshima, Koji Iwanuma, Katsumi Inoue, and Oliver Ray. „SOLAR: An automated deduction system for consequence finding“. In: *AI Communications* 23.2-3 (2010), pp. 183–203.

[Nic+13] Iulia Nica, Ingo Pill, Thomas Quaritsch, and Franz Wotawa. „The Route to Success - A Performance Comparison of Diagnosis Algorithms“. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Bejing, China, 2013, pp. 1039–1045.

[Nie93] Jakob Nielsen. „Iterative user-interface design“. In: *Computer* 26.11 (1993), pp. 32–41.

[NM92] Hwee Tou Ng and Raymond J Mooney. „Abductive plan recognition and diagnosis: A comprehensive empirical evaluation“. In: *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc. 1992, pp. 499–508.

[NW12] Iulia Nica and Franz Wotawa. „ConDiag-computing minimal diagnoses using a constraint solver“. In: *International Workshop on Principles of Diagnosis*. 2012, pp. 185–191.

[NW97] P Pandurang Nayak and Brian C Williams. „Fast context switching in real-time propositional reasoning“. In: *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*. AAAI Press. 1997, pp. 50–56.

[Nyb11] Mattias Nyberg. „A generalized minimal hitting-set algorithm to handle diagnosis with behavioral modes“. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41.1 (2011), pp. 137–148.

[NZ08] Gustav Nordh and Bruno Zanuttini. „What makes propositional abduction tractable“. In: *Artificial Intelligence* 172.10 (2008), pp. 1245–1284.

[Ovc+13] Ekaterina Ovchinnikova, Andrew S Gordon, and Jerry Hobbs. „Abduction for discourse interpretation: a probabilistic framework“. In: *Proceedings of the Joint Symposium on Semantic Processing*. 2013, pp. 42–50.

[Pau93] Gabriele Paul. „Approaches to abductive reasoning: an overview“. In: *Artificial intelligence review* 7.2 (1993), pp. 109–152.

[PD01] James D Park and Adnan Darwiche. „Approximating MAP using local search“. In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 2001, pp. 403–410.

[PD95] Michael Pecht and Abhijit Dasgupta. „Physics-of-failure: an approach to reliable product development“. In: *Journal of the IES* 38.5 (1995), pp. 30–34.

[Pea88] Judea Pearl. „Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference“. In: (1988).

[Pei+16] Bernhard Peischl, Ingo Pill, and Franz Wotawa. „Abductive Diagnosis based on Modelica Models“. In: (2016), pp. 1–9.

[Pei74] Charles Sanders Peirce. *Collected papers of charles sanders peirce*. Vol. 5. Harvard University Press, 1974.

[Pil+11] Ingo Pill, Thomas Quaritsch, and Franz Wotawa. „From conflicts to diagnoses: An empirical evaluation of minimal hitting set algorithms“. In: *22nd International Workshop on the Principles of Diagnosis*. 2011, pp. 203–210.

[Poo+87] David Poole, Randy Goebel, and Romas Aleliunas. „Theorist: A logical reasoning system for defaults and diagnosis“. In: *The Knowledge Frontier*. Springer, 1987, pp. 331–352.

[Pop73] Harry E Pople. „On the mechanization of abductive logic“. In: *Proceedings of the 3rd international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc. 1973, pp. 147–152.

[Por+04] Luigi Portinale, Diego Magro, and Pietro Torasso. „Multi-modal diagnosis combining case-based and model-based reasoning: a formal and experimental analysis“. In: *Artificial Intelligence* 158.2 (2004), pp. 109 –153. ISSN: 0004-3702. URL: `http://www.sciencedirect.com/science/article/pii/S0004370204000839`.

[PQ12] Ingo Pill and Thomas Quaritsch. „Optimizations for the Boolean approach to computing minimal hitting sets“. In: *Proceedings of the 20th European Conference on Artificial Intelligence*. IOS Press. 2012, pp. 648–653.

[PR89] Yun Peng and James A Reggia. „A connectionist model for diagnostic problem solving“. In: *Systems, Man and Cybernetics, IEEE Transactions on* 19.2 (1989), pp. 285–298.

[PR90] Yun Peng and James A Reggia. *Abductive inference models for diagnostic problem-solving*. Springer, 1990.

[PT02] Chris J Price and Neil S Taylor. „Automated multiple failure FMEA“. In: *Reliability Engineering & System Safety* 76.1 (2002), pp. 1–10.

[PW03] Bernhard Peischl and Franz Wotawa. „Computing diagnosis efficiently: A fast theorem prover for propositional Horn theories“. In: *Proceedings of the 14th International Workshop on Principles of Diagnosis*. 2003, pp. 175–180.

[Qiu+12] Yingning Qiu, Yanhui Feng, Peter Tavner, et al. „Wind turbine SCADA alarm analysis for improving reliability“. In: *Wind Energy* 15.8 (2012), pp. 951–966.

[QP14] Thomas Quaritsch and Ingo Pill. „Pymbd: A library of mbd algorithms and a light-weight evaluation platform“. In: *Proceedings of the 25th International Workshop on Principles of Diagnosis*. 2014, pp. 1–5.

[Qua14] Thomas Quaritsch. „Diagnosis of LTL Specifications using Consistency-oriented Model-based Reasoning“. PhD thesis. Graz University of Technology, 2014.

[Qui55] Willard V Quine. „A way to simplify truth functions“. In: *American mathematical monthly* (1955), pp. 627–631.

[Rad+93] LWMM Rademakers, A Seebregts, and B van Den Horn. *Reliability analysis in wind engineering*. Netherlands Energy Research Foundation ECN, 1993.

[RD97] Antoine Rauzy and Yves Dutuit. „Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within Aralia“. In: *Reliability Engineering & System Safety* 58.2 (1997), pp. 127–144.

[Rei87] Raymond Reiter. „A theory of diagnosis from first principles“. In: *Artificial Intelligence* 32.1 (1987), pp. 57–95.

[Reu17] Reuters. *Atlanta airport power outage cost Delta Air Lines up to $50 million*. 2017. URL: `https://www.reuters.com/article/us-atlanta-airport-delta-air/atlanta-airport-power-outage-cost-delta-air-lines-up-to-50-million-idUSKBN1EE2LC`.

[RH04] Marvin Rausand and Arnljot Høyland. *System reliability theory: Models, statistical methods, and applications*. Vol. 396. John Wiley & Sons, 2004.

[Ric76] John R Rice. „The algorithm selection problem“. In: *Advances in computers* 15 (1976), pp. 65–118.

[Rob+13] C Robinson, E Paramasivam, E Taylor, A Morrison, and E Sanderson. *Study and development of a methodology for the estimation of the risk and harm to persons from wind turbines*. Tech. rep. Technical report, Health and Safety Executive, 2013.

[RS15] Enno Ruijters and Mariëlle Stoelinga. „Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools“. In: *Computer science review* 15 (2015), pp. 29–62.

[Rym94] Ron Rymon. „An SE-tree-based prime implicant generation algorithm“. In: *Annals of Mathematics and Artificial Intelligence* 11.1 (1994), pp. 351–365.

[Sac+00] Martin Sachenbacher, Peter Struss, and Reinhard Weber. *Advances in design and implementation of OBD functions for diesel injection based on a qualitative approach to diagnosis*. Tech. rep. SAE Technical Paper, 2000.

[Sai+16] Paul Saikko, Johannes P Wallner, and Matti Järvisalo. „Implicit hitting set algorithms for reasoning beyond NP“. In: *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*. AAAI Press. 2016, pp. 104–113.

[Sch+13] Meik Schlechtingen, Ilmar Ferreira Santos, and Sofiane Achiche. „Wind turbine condition monitoring based on SCADA data using normal behavior models. Part 1: System description“. In: *Applied Soft Computing* 13.1 (2013), pp. 259 –270. ISSN: 1568-4946. URL: `http://www.sciencedirect.com/science/article/pii/S1568494612003821`.

[Sch16] Peter Schüller. „Modeling variations of first-order Horn abduction in answer set programming“. In: *Fundamenta Informaticae* 149.1-2 (2016), pp. 159–207.

[SD89] Peter Struss and Oskar Dressier. „Physical negation: integrating fault models into the general diagnostic engine“. In: *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc. 1989, pp. 1318–1323.

[SDV01] Laurent Simon and Alvaro Del Val. „Efficient consequence finding“. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence-Volume 1*. Morgan Kaufmann Publishers Inc. 2001, pp. 359–365.

[Shc+15] Kostyantyn Shchekotykhin, Dietmar Jannach, and Thomas Schmitz. „MERGEXPLAIN: fast computation of multiple conflicts for diagnosis“. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press. 2015, pp. 3221–3228.

[SI00] Chiaki Sakama and Katsumi Inoue. „Abductive logic programming and disjunctive logic programming: their relationship and transferability“. In: *The Journal of Logic Programming* 44.1 (2000), pp. 75–100.

[SK96] Bart Selman and Henry Kautz. „Knowledge compilation and theory approximation“. In: *Journal of the ACM (JACM)* 43.2 (1996), pp. 193–224.

[Sla+70] James R Slagle, Chin-Liang Chang, and Richard CT Lee. „A new algorithm for generating prime implicants“. In: *IEEE transactions on Computers* 100.4 (1970), pp. 304–310.

[Sol+17] Marc Solé, Victor Muntés-Mulero, Annie Ibrahim Rana, and Giovani Estrada. „Survey on Models and Techniques for Root-Cause Analysis“. In: *arXiv preprint arXiv:1701.08546* (2017).

[Sta10] Switzerland International Organization for Standardization Geneva. *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.ISO 9241-210:2010*. Norm. 2010.

[Str+96] Peter Struss, Andreas Malik, and Martin Sachenbacher. „Case studies in model-based diagnosis and fault analysis of car-subsystems“. In: *Proceedings of the 1st International Workshop Model-based Systems and Qualitative Reasoning*. 1996, pp. 17–25.

[Str08] Peter Struss. „Model-based Problem Solving“. In: *Handbook of Knowledge Representation*. Ed. by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. Oxford: Elsevier Science, 2008. Chap. 10, pp. 395–465.

[SW01] Markus Stumptner and Franz Wotawa. „Diagnosing tree-structured systems“. In: *Artificial Intelligence* 127.1 (2001), pp. 1–29.

[Tag05] Nancy R Tague. *The quality toolbox*. Vol. 600. ASQ Quality Press Milwaukee, 2005.

[Taz+17] Nacef Tazi, Eric Châtelet, and Youcef Bouzidi. „Using a Hybrid Cost-FMEA Analysis for Wind Turbine Reliability Analysis“. In: *Energies* 10.3 (2017), p. 276.

[Tch+14] Pierre Tchakoua, René Wamkeue, Mohand Ouhrouche, et al. „Wind turbine condition monitoring: State-of-the-art review, new trends, and future challenges“. In: *Energies* 7.4 (2014), pp. 2595–2630.

[Tis67] Pierre Tison. „Generalization of consensus theory and application to the minimization of boolean functions“. In: *Electronic Computers, IEEE Transactions on* 4 (1967), pp. 446–456.

[TMM98] Louise Travé-Massuyès and Robert Milne. „Gaps between research and industry related to model based and qualitative reasoning“. In: *Proceedings of the European workshop on Model based systems and qualitative reasoning*. 1998, pp. 54–57.

[Tse70] Gregory Tseitin. „On the complexity of proofs in propositional logics“. In: *Seminars in Mathematics*. Vol. 8. 1970, pp. 466–483.

[TSM16] Houari Toubakh and Moamar Sayed-Mouchaweh. „Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters“. In: *Neurocomputing* 171 (2016), pp. 1496–1516.

[Val99] Alvaro del Val. „A new method for consequence finding and compilation in restricted languages“. In: *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*. American Association for Artificial Intelligence. 1999, pp. 259–264.

[Vog+14] Gregory W Vogl, Brian A Weiss, and M Alkan Donmez. „Standards for prognostics and health management (phm) techniques within manufacturing operations“. In: *Annual Conference of the Prognostics and Health Management Society, September*. 2014, pp. 576–588.

[VP08] Alberto Venturini and Gregory Provan. „Incremental algorithms for approximate compilation“. In: *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*. AAAI Press. 2008, pp. 1495–1498.

[Wei+08] Wanxia Wei, Chu Min Li, and Harry Zhang. „Switching among Non-Weighting, Clause Weighting, and Variable Weighting in Local Search for SAT“. In: *Proceedings of the 14th international conference on Principles and Practice of Constraint Programming*. Springer-Verlag. 2008, pp. 313–326.

[Wil+10] Michael Wilkinson, B Hendriks, F Spinato, et al. „Methodology and results of the ReliaWind reliability field study“. In: *European Wind Energy Conference and Exhibition 2010, EWEC 2010*. Vol. 3. Sheffield. 2010, pp. 1984–2004.

[Wil45] Frank Wilcoxon. „Individual comparisons by ranking methods“. In: *Biometrics bulletin* 1.6 (1945), pp. 80–83.

[Wit+11] Ian H. Witten, Eibe Frank, and Mark A Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2011. ISBN: 0123748569, 9780123748560.

[WK+14] Fang Wei-Kleiner, Zlatan Dragisic, and Patrick Lambrix. „Abduction framework for repairing incomplete EL ontologies: complexity results and algorithms“. In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press. 2014, pp. 1120–1127.

[WN96] Brian C Williams and P Pandurang Nayak. „A model-based approach to reactive self-configuring systems“. In: *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 2*. AAAI Press. 1996, pp. 971–978.

[Wot+09] Franz Wotawa, Ignasi Rodriguez-Roda, and Joaquim Comas. „Abductive Reasoning in Environmental Decision Support Systems“. In: *Proceedings of the Workshop on Artificial Intelligence Applications on Environmental Protection*. 2009, pp. 270–279.

[Wot+10] Franz Wotawa, Ignasi Rodriguez-Roda, and Joaquim Comas. „Environmental decision support systems based on models and model-based reasoning“. In: *Environmental Engineering & Management Journal (EEMJ)* 9.2 (2010).

[Wot01] Franz Wotawa. „A variant of Reiter's hitting-set algorithm“. In: *Information Processing Letters* 79.1 (2001), pp. 45–51.

[Wot09] Franz Wotawa. „On the use of abduction as an alternative to decision trees in environmental decision support systems“. In: *Complex, Intelligent and Software Intensive Systems, 2009. CISIS'09. International Conference on*. IEEE. 2009, pp. 1160–1165.

[Wot14] Franz Wotawa. „Failure Mode and Effect Analysis for Abductive Diagnosis“. In: *Proceedings of the International Workshop on Defeasible and Ampliative Reasoning*. 2014, pp. 1–13.

[WS01] Franz Wotawa and Markus Stumptner. „Modellbasierte Diagnose—Überblick und technische Anwendung“. In: *e & i Elektrotechnik und Informationstechnik* 118.7 (2001), pp. 360–366.

[Xu+08] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. „SATzilla: portfolio-based algorithm selection for SAT“. In: *Journal of artificial intelligence research* 32 (2008), pp. 565–606.

[You+00] Jia-Huai You, Li Yan Yuan, and Randy Goebel. „An abductive approach to disjunctive logic programming“. In: *The Journal of Logic Programming* 44.1-3 (2000), pp. 101–127.

[ZL+12] Yang Zhi-Ling, Wang Bin, Dong Xing-Hui, and LIU Hao. „Expert system of fault diagnosis for gear box in wind turbine“. In: *Systems Engineering Procedia* 4 (2012), pp. 189–195.

[ZM07] A. S. Zaher and S.D.J. McArthur. „A Multi-Agent Fault Detection System for Wind Turbine Defect Recognition and Diagnosis“. In: *2007 IEEE Lausanne Power Tech* (July 2007), pp. 22–27. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4538286.