Model-Based Reconfiguration: Diagnosis and Recovery

Judy Crow and John Rushby

Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025 USA

NASA Contractor Report 4596, May 1994

Abstract

We extend Reiter's general theory of model-based diagnosis to a theory of fault detection, identification, and reconfiguration (FDIR). The generality of Reiter's theory readily supports an extension in which the problem of reconfiguration is viewed as a close analogue of the problem of diagnosis. Using a reconfiguration predicate **rcfg** analogous to the abnormality predicate **ab**, we derive a strategy for reconfiguration by transforming the corresponding strategy for diagnosis. There are two obvious benefits of this approach: first, algorithms for diagnosis can be exploited as algorithms for reconfiguration; second, we have a theoretical framework for an integrated approach to FDIR. As a first step toward realizing these benefits, we show that a class of diagnosis engines can be used for reconfiguration and we discuss algorithms for integrated FDIR. We argue that integrating recovery and diagnosis is an essential next step if this technology is to be useful for practical applications.

Contents

List of Figures

Chapter 1

Introduction

Automated diagnosis has been one of the more fruitful applications of AI, with potential significance for domains in which diagnosis of complex physical systems must proceed while the system is operative and testing opportunities are limited by operational considerations. In air- and spacecraft applications, for example, maintaining a system in a safe operating state during diagnosis clearly precludes certain tests and/or additional measurements; it is impossible to add additional sensors to an orbiting spacecraft or to run tests which could render the vehicle inoperative. Operative diagnosis (*cf.* Chapter 2, Section 2.2) thus differs from what is generally referred to as *maintenance diagnosis*, where faults are diagnosed in the shop rather than in the field. However, while it may be interesting and even useful to identify the faults in a malfunctioning system in either a maintenance or operative context, the real problem is usually to fix the system so that it can continue its mission. Thus in many applications, diagnosis is only part of a larger problem known as Fault Detection, Identification, and Reconfiguration (FDIR)[1]. Surprisingly, despite the interest in diagnosis, there has been relatively little work on the foundations of recovery and reconfiguration, and virtually none on the problem of integrated FDIR—although the practical benefits of an integrated approach could be considerable, especially when knowledge of available reconfigurations is used to reduce the cost and increase the accuracy and utility of diagnosis.

[1]FDIR can stand for Fault Detection, Identification, and Reconfiguration (our preferred interpretation) or Fault Diagnosis, Isolation, and Recovery, or various combinations thereof.

1.1 The Approach

The practical motivation for our work derives from systems such as airplanes and spacecraft, which typically possess considerable redundancy in the form of backup systems, as well as degraded operating modes. In this paper we present a theory of reconfiguration for such systems as an analogue of Reiter's model-based theory of diagnosis [40]. We chose Reiter's theory as a point of departure because it provides a formal characterization of diagnosis shared to some extent by most of the model-based systems described in the literature, including DART [21], GDE [12] and its descendants [13, 23], and the work of Davis [7]. Our approach follows from two basic insights: first, the generality of Reiter's theory of diagnosis makes it applicable to other domains; second, a productive analogy exists between the problem of diagnosis and that of reconfiguration. Diagnosis is the problem of identifying components whose abnormality is sufficient to explain an observed malfunction. Similarly, reconfiguration can be viewed as the problem of identifying components whose reconfiguration is sufficient to restore acceptable behavior. Two potential benefits result from characterizing reconfiguration as an extension of Reiter's theory of diagnosis in this way: first, we can exploit algorithms for diagnosis as algorithms for reconfiguration, and second, we have a unified framework that should facilitate the development of an integrated theory of FDIR.

1.2 Why FDIR?

We view the limited focus of extant work on automated fault diagnosis, whether rule-based or model-based, as a serious drawback to its practical applicability. In many practical applications, fault diagnosis is only part of the problem; the larger problem is FDIR. Thus classical approaches to fault diagnosis, which simply identify the fault, solve only half the problem of automated FDIR. Reconfiguration and recovery, the other half of the problem, is typically either ignored, reduced to a set of preplanned procedures (which are inherently at odds with the expressed intent of model-based approaches) or handled as a planning problem distinct from the original diagnosis problem. In contrast, we believe that the most effective approaches will be those that consider FDIR as an integrated problem, in which diagnosis and recovery are solved in concert. Some of the potential benefits of an integrated approach to FDIR are:

- A single computational engine can be used for both diagnosis and reconfiguration.
- A significant reduction of the search space can be achieved: only those diagnoses that require different reconfigurations need be distinguished, and the number of possible reconfigurations is typically much smaller than the number of diagnoses.
- Temporary reconfigurations can be used to discriminate among competing diagnoses: e.g., does the symptom disappear when we switch to a backup system?
- Application to domains such as real-time operative systems [1] becomes more relevant, accommodating, for example, the requirement to place the system in a safe state even before a solid diagnosis is available.
- A broader context is provided for both diagnosis and recovery, in which potential consequences of misdiagnosed faults and incorrect recovery actions can be properly evaluated, and resources effectively apportioned.

As noted above, our efforts have focused primarily on defining an effective basis for integrated FDIR. The theory we develop in this paper does not realize these benefits; our objective here is to propose a characterization of reconfiguration that will promote this goal of effective integration. Furthermore, although we have not approached the problem of FDIR in explicitly operative terms, our approach is fundamentally operative; reconfiguration or recovery is a significant component of FDIR precisely because it enables a system to correct or compensate for abnormal behavior, i.e., to continue operating in a specifiably acceptable manner. Similarly, although we have not accommodated real-time factors, the generality of our approach suggests that it should be possible to factor in real-time constraints.

1.3 Organization

The organization of this report is as follows. Chapter 2 provides a survey of current approaches to the problem of diagnosis, including a brief account of research in real-time and operative diagnosis, as well as related work in reconfiguration and recovery. Chapter 3 develops the context for our work, presenting a fairly detailed discussion of Reiter's theory of diagnosis. We move to an account of our extensions to Reiter's theory in Chapter 4, where we develop our characterization of reconfiguration. Chapter 5 consists of a series of examples illustrating the ideas in the two previous chapters. In Chapter 6, we formalize the correspondence between diagnosis and reconfiguration, defining a mapping between a class of diagnosis engines and reconfiguration engines that raises the possibility of an integrated FDIR engine. Chapter 7 examines potential limitations of our approach, focusing primarily on issues of minimality, consistency, and entailment. The final chapter summarizes our work and suggests an agenda for future research.

Chapter 2

Survey of Diagnosis

Automated diagnosis is inherently interdisciplinary, both in its techniques and its applications. A representative but by no means exhaustive list of research problems includes mathematical modeling and simulation; logics for and theories of causal, temporal, and qualitative reasoning; constraint systems, truth maintenance systems, and knowledge representation; and sensor optimization and validation. Historically, the main application domains for research in automated diagnosis have been medicine/physiology and analog/digital circuits, although there certainly exist applications or application prototypes in a variety of other domains including power plant diagnosis, hydraulic systems, and aircraft subsystems.

Automated diagnosis proceeds from two primary assumptions:

- The objective is to diagnose malfunctions, not design errors.

- Tests are more expensive than computation and misdiagnoses are more expensive than tests.

Accordingly, the goal of automated diagnosis is to find an effective balance between coverage, accuracy, specificity, and efficiency.

Historically, the field has moved from systems which exploit preenumerated, device-specific, symptom-fault associations encoded either as rule bases or fault dictionaries, to systems which "reason" from basic principles about causality and from structural and behavioral device models. We briefly explore this evolution below, and then turn to issues in real-time and operative diagnosis.

2.1 The Evolution of Knowledge-Based Fault Diagnosis

2.1.1 Traditional Techniques

Traditional approaches to diagnosis can generally be characterized as one of the following:

- Diagnostics: test programs typically run at the end of the manufacturing line to verify that a device functions "correctly"
- Prespecified Fault Models: fault dictionaries or preenumerated models of symptom-fault behavior
- Rule-Based Systems: encoded empirical association of symptoms and faults accumulated by expert troubleshooters in a given domain
- Decision Trees: convention for recording the diagnostic process, i.e., the tests and conclusions required for a particular diagnostic strategy

The first and last approaches, traditional diagnostics and decision trees, are included primarily for historical reasons; they have contributed only marginally to subsequent theories of diagnosis. Davis [7, p. 360] notes that traditional diagnostics actually do verification rather than diagnosis; i.e., they verify that a device correctly executes all intended behaviors rather than diagnosing its misbehavior, although they apparently have been used for diagnosis as well as verification. Decision trees are useful for codifying diagnostic strategies, but have no explanatory power: they do not provide explanation or insight into the diagnosis.

However, fault dictionaries and rule-based systems are effective diagnostic techniques.[1] Both preenumerated fault models and rule-based systems are symptom based, i.e., they encode models of preselected symptom-fault associations.[2] Examples of early symptom-based systems include ABEL which uses a behavioral model to represent the causal relation(s) between physiological events in the body [34], and IDS which diagnoses dependent

[1]Davis [7, p. 361] reports that a large percentage of all faults in a digital circuit can be *detected*, although not diagnosed, by using a fault model (dictionary) to check for stuck-ats or faults in which a node in the circuit always exhibits the value 0(1).

[2]For example, a fault dictionary is generated by simulating the behavior of a given set of components over a preselected list of anticipated faults. The resulting list of fault-symptom pairs is inverted to provide a dictionary which indexes from symptoms to one or more faults consistent with a given misbehavior.

failures by representing the behavior of analog circuit components given out-of-range inputs [33]. ABEL, IDS, and other relatively early, symptom-based systems such as MYCIN [42], INTERNIST-I [36], and System D [38], contributed several ideas subsequently pursued in the context of model-based diagnosis, including techniques for reducing the number of active candidate hypotheses, hierarchic models, approaches such as set covers for dealing with multiple independent diseases/faults, and sequential diagnosis, a probabilistic (Bayesian) method for choosing a next measurement most likely to lead to discovery of the actual candidate with a minimum number of subsequent measurements.

Symptom-based models using preenumerated fault models exhibit three desirable properties: they work even if the exact mechanisms of a device are not well understood; they work if the device is complex, but fails in a small number of predictable ways; and they are effective at reducing a large space of possibilities to a small set of reasonable hypotheses. Their undesirable properties derive from a strong device dependence; introducing a new device/disease typically requires a new rule set, and even minor changes to a device can invalidate an existing rule base. Furthermore, it takes a nontrivial amount of time to acquire sufficient experience with a given device/disease to expose diagnostic patterns and build the rule base; this is potentially a serious drawback for applications such as electronics which exhibit increasingly shorter design cycles. Davis and Hamscher [8, p. 304] also argue that rules are an inappropriate representation for diagnostic systems because they don't readily express structural and behavioral information. A further drawback which derives at least in part from device dependence is the inability to deal with unanticipated faults.

Model-based techniques can be viewed as a response to these limitations, where by model-based we refer specifically to techniques which exploit structural and behavioral models for diagnosis. Model-based diagnosis differs from the previously discussed symptom-based approaches, which also employ models, with respect to the type and use of models; model-based diagnosis uses general inferencing mechanisms to focus on the relation between structural and behavioral models. Model-based diagnosis is also referred to as "diagnosis from first principles," reflecting precisely this emphasis on structural and behavioral models in conjunction with general causal principles.

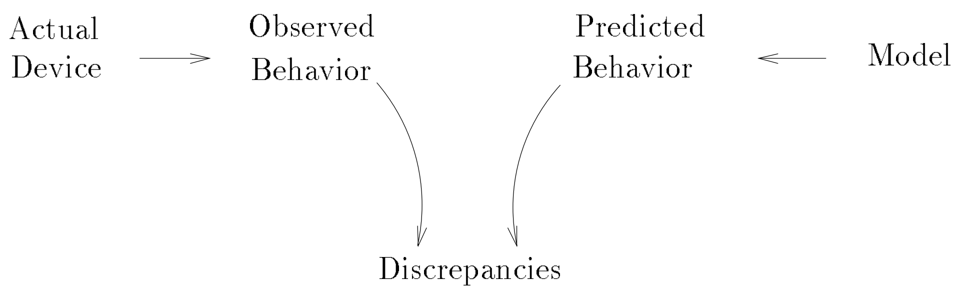

Figure 2.1: Characterization of model-based diagnosis

2.1.2 Model-Based Techniques

The basic paradigm of model-based diagnosis can be characterized as the interaction of observation and prediction; the behavior of a physical device is observed and compared with the predicted behavior of a model of the same device. The event(s) of interest are those in which the two differ, i.e., there is a *discrepancy* between observation and prediction. Figure 2.1, reproduced from [25], represents this paradigm schematically.

Model-based diagnosis assumes that if the model is correct, all discrepancies between observation and prediction derive from and can be traced to faults in the device. Note the comparatively comprehensive definition of fault; unlike the preselected fault lists employed by traditional techniques, model-based diagnosis defines a fault as any discrepancy between observed and predicted behavior; of course, the assumption that the model is correct is itself open to question. Modulo limitations of the model, this approach potentially encompasses both novel and anticipated faults, and has the further advantages that while the model, which can be produced at design time, is obviously device specific, the diagnosis is handled by a general, device independent program. Davis and Hamscher [8, p. 309] identify three largely self-explanatory tasks within the model-based framework: hypothesis generation, hypothesis testing, and hypothesis discrimination, which are generally interleaved in system implementations, and note that model-based systems can be distinguished with respect to the kinds and amounts of knowledge used for each task. Hamscher and Patil [25, p. MA1-150] take a broader and somewhat different view, identifying five tasks: modeling, prediction, candidate generation, discrimination, and elaboration. The mapping between the Davis/Hamscher task distinctions and those of Hamscher/Patil is shown in

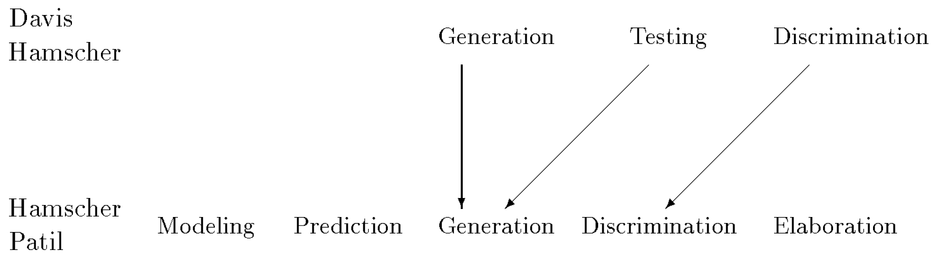

Figure 2.2: Comparison of diagnosis task distinctions

Figure 2.2. The differences arise from the fact that Davis/Hamscher do not explicitly identify modeling and prediction as tasks, while Hamscher/Patil collapse the tasks of test and generation and explicitly represent the notion of elaboration (control).

Using Hamscher and Patil's task discrimination, some of the key issues associated with each of the tasks are summarized below.

- Modeling: Models are approximations; artifacts of a device not captured in the model may or may not have consequences for diagnosis. This is the familiar tradeoff between completeness and complexity. Generally accepted strategies include the use of hierarchical models and, wherever possible, models isomorphic to the structure of the mechanism being modeled. There are also well-known, though not universally respected dictums such as Brown, Burton and de Kleer's [4] "no-function-in-structure".[3]

- Prediction: Prediction is expensive; the tradeoff here is precision versus cost. In the context of model-based diagnosis, prediction is more than traditional simulation; numerical simulation is not suited to the low-resolution, partial information found in diagnosis. Furthermore, traditional simulation typically provides a single, precise projection

[3] "No-function-in-structure" refers to the dictum that component behaviors should be defined independent of their use in a particular device. For example, a switch is defined in terms of position and resistance—resistance is low when the switch is closed, high when it is open—independent of its function in a particular circuit. A consequence is that behavior is independent of design location.

rather than a set of alternative possibilities and associated assumptions ("reasons"). Hamscher and Patil [25, p. MA1-90–91] cite four approaches used to optimize precision with respect to cost: stepwise simulation in numeric models, event-driven simulation in discrete models, qualitative simulation, and techniques for temporal abstraction.

- Candidate Generation: At each step the number of potential candidates can be large; the issue is constraining the indiscriminacy. Ideally, the generator is complete, i.e., produces all plausible candidate hypotheses, nonredundant, i.e., generates each candidate only once, and informed, i.e., produces a few, ultimately correct hypotheses. There are several approaches to candidate generation, including those enumerated below.

  - *Upstream tracing*, the simplest and least constrained strategy, considers any component connected to and upstream of a given location in the system to be a potential suspect. There is also a related strategy referred to as corroboration or direct exoneration in which anything upstream of a good value is assumed innocent. Direct exoneration can be viewed as the dual of upstream tracing and works only in the absence of masked faults, which are, of course, tricky to rule out. Furthermore, while failing to exonerate an innocent suspect can result in unnecessary testing, mistakenly exonerating a faulty component has far more serious consequences. However, when used judiciously, corroboration can be a productive technique. SOPHIE [4, p. 124] uses corroborations as well as conflicts (*cf.* below) for candidate elimination.

  - *Prediction-constrained tracing* exploits knowledge about intended component behaviors to expose suspect components. This approach typically assumes a simulator which propagates reasons as well as values. SOPHIE [4] is one of the earliest examples.

  - *Constraint suspension* checks consistency of a suspect against observed behaviors as follows. The behavior of each component is modeled as a set of constraints. The set of constraints associated with a suspect is suspended, i.e., removed from the constraint network, and the modified network run to quiescence. If the network does not encounter an inconsistency, the current suspect is consistent with (i.e., could be responsible for) the given observations. Constraint suspension was proposed by Davis and used in HT [7].

- ◦ *Conflict detection* uses an assumption-based truth maintenance system (ATMS) which propagates reasons (sets of assumptions) as well as predicted values. Observations (e.g., input, output values) have no assumptions. Values are traced through the model, with contradictory predictions yielding a *conflict*, a set of all assumptions underlying each of the conflicting predictions. When the propagation terminates, all conflicts are collected and a set of candidates consistent with the collection of conflicts generated. Since this process is equivalent to the problem of generating set covers, which is exponential in the worst case, only minimal conflicts and minimal candidates are generated, where a minimal conflict/candidate is one which contains no sets which are also conflicts/candidates. This technique, which generates both single and multiple fault candidates, was developed by de Kleer and Williams and used in GDE [12].

- Discrimination: Discriminating among candidates involves acquiring additional information; the issue is containing the lookahead costs. There are basically two approaches: probing, which is noninvasive and involves making additional observations, and testing, which is invasive and involves perturbing the state of the device. Hamscher and Patil [25, p. MA1-125] note that probing techniques are $O(2^n n^{2k})$ for k-level lookahead with n components and that the problem of test generation is intractable.

  - ◦ Probing: probing strategies are typically variants of the guided probe: starting at a discrepancy, the malfunction is traced upstream to a component whose inputs are correct, but whose output is incorrect. This technique can be extended to use information about component behavior in order to reduce the number of probes, but still requires a linear time search. Additional information such as device topology can yield probe points which ideally split the search space at each step, particularly in cases where there is an obviously most informative probe. In the case of several equally informative probes, failure probabilities can be used. There are also more sophisticated techniques which attempt to optimize probe selection, such as sequential diagnosis, the information theoretic approach taken in GDE [12] that uses an evaluation function based on the notion of minimum entropy and exploits the rich context maintained by the ATMS to try to identify the smallest effective sequence of measurements.

  - Testing: hypothesis discrimination via testing potentially produces new symptoms and suspects, which must be reconciled with the existing set of candidates (typically the intersection or set cover of the old and new suspects). Test selection must be optimized, e.g., as a function of test cost, coverage, and specificity; or, if the set of possible tests is unknown or intractably large, test generation must be effectively confronted using planning or knowledge-based techniques [25, p. MA1-124] or some other means of constraining the problem.

- Elaboration: Automated diagnosis necessarily involves a number of control issues, sucha as whether the next step in the diagnosis should be generation or discrimination and at what level/layer of the model and/or in which model. Strategies for candidate elaboration include fault envisionment, using fault models to eliminate candidates (GDE [12]); hierarchic diagnosis, reasoning at the most abstract level in the hierarchy and descending only when necessary to check suspect component(s) (DART [21] and HT [7]); layered models, enumerating categories of failure and producing an ordered layering of fault types based on a given metric, such as failure frequency, to guide candidate generation (HT).

This discussion of diagnosis tasks has highlighted key developments and systems in the history of model-based diagnosis. One further development is the growth of hybrid systems, i.e., systems which incorporate elements of both symptom- and model-based strategies. Abbott [2] is an example; using Davis' approach which exploits both physical and functional models as well as layered models to partition the search space into a small number of disjoint classes, Abbott develops a notion of operative diagnosis which involves incremental hypothesis construction and reasoning about fault propagation in an active system. We refer again to Abbott's work in the context of real-time and operative diagnosis in Section 2.2. Struss and colleagues [44] supplement model-based diagnosis with explicit fault models and also exemplify the trend toward a synthesis of symptom- and model-based diagnosis. We next briefly consider two as yet unmentioned developments: qualitative models and a logical perspective on model-based reasoning.

2.1.3 Qualitative Physics

Qualitative Physics is a subfield of Artificial Intelligence (AI) concerned with representing and reasoning about the physical world. Forbus identifies the goal of qualitative physics as an attempt to capture "...both the common-sense knowledge of the person on the street and the tacit knowledge underlying the quantitative knowledge used by engineers and scientists." [17, p. 11] Diagnosis represents one of many applications of this research in qualitative representation and reasoning. AI literature has traditionally distinguished *shallow* models/reasoning, i.e., the type of reasoning/models used in expert systems, where conclusions are drawn directly from observable or directly represented features of the domain, from *deep* models/reasoning in which the desired result is drawn from underlying mechanisms whose parameters are not necessarily directly observable. Qualitative causal models are deep models, but differ from other deep models in focusing on qualitative descriptions capable of representing partial knowledge of structure and behavior. These descriptions are generated by examining the physical structure of a device and deriving a set of constraint equations for the relevant structural relationships. Possible behaviors of the system can be predicted by qualitative simulation from the constraint equations and an initial state. The qualitative behavioral description may be used in conjunction with the qualitative structural description to explain a set of observations, such as the misbehavior of a physical system. There have been three major approaches to the derivation of constraint equations. As suggested earlier, de Kleer and Brown [9] and Williams [45] use a device-centered ontology; physical systems are described in terms of components and connections. Approaching the problem from the perspective of naive physics, Forbus uses a process ontology; physical systems are described as a set of active processes [16]. Kuipers [29], on the other hand, treats the constraint equations as given, and focuses on issues in qualitative simulation. The central inference common to all three approaches is qualitative simulation, i.e., derivation of a description of the behavior of a system from the qualitative structural description *qua* constraint equations. Given our previous discussion of the mechanisms of model-based diagnosis, the relevance of qualitative causal models and qualitative reasoning should be clear.

2.1.4 The Role of Logic in Model-Based Reasoning

Logic has also been used as a representation language for model-based diagnosis. We briefly consider two of these uses. Genesereth, one of the first to use first-order logic to represent systems (DART) [21], employed a resolution style theorem prover to compute candidate faults and to discriminate among competing diagnoses. Reiter [40] subsequently extended and generalized some of Genesereth's results in a formal theory of diagnosis which captures several of the previously discussed approaches to diagnosis including de Kleer's conflict sets [12], de Kleer and Williams' characterization of diagnoses [12], Davis' candidate generation procedure [7], and Reggia, Nau, and Wang's generalized set covering (GSC) model [38]. Reggia *et al.* refer to the GSC approach to diagnosis as "abductive diagnostic inference." The formal relation between Reiter's theory of diagnosis and abductive inference has also been drawn by Cox and Pietrzykowski [6]. Finally, Reiter observes that diagnostic reasoning is nonmonotonic, and relates his theory of diagnosis to default logic, suggesting yet another connection between diagnosis and developments in classical and nonclassical logics.

2.2 Real-Time and Operative Diagnosis

In this section we turn to knowledge-based fault diagnosis with an emphasis on *operative* and *real-time* diagnosis, where operative diagnosis refers to diagnosis of physical systems in operation [2] and real-time diagnosis refers to time-critical diagnosis. Note that the notions of operative and real-time are disjoint, and further that the notion of real-time does not necessarily imply a focus on *time*.[4] Research in operative and/or real-time diagnosis can be broadly classified as reorganizing and refining algorithms, architectures, and tools to explicitly accommodate real-time and operative constraints, or as optimizing knowledge-based systems for real-time applications through various compilation strategies.

Examples of reorganization strategies include the ABE/RT toolkit, a set of design, development, and experimentation tools for building time-critical intelligent systems which was initially developed for the Lockheed Pilot's Associate application [30]. The three distinct, but interlocking languages offered in the toolkit allow explicit representation of hierarchical structures, timeliness, and resource allocation requirements. Another example is Abbott's DRAPHYS system [1,2] for operative diagnosis of aircraft subsystems, which offers graceful degradation in the presence of novel faults by exploiting

[4] For example, in a brief note Schneider questions the "implicit belief ... that time is fundamental to real-time programs," suggesting instead new paradigms such as synchronizing asynchronous processes [41].

abstraction mechanisms, multiple fault classes, and incremental hypothesis construction supported by simulation using several models of fault propagation behavior. Abbott focuses on operative diagnosis, largely ignoring real-time issues. Further examples of the reorganization/refinement strategy include the work of Korf on adapting classic search algorithms for real-time problem solving [28]; studies of architectures for real-time problem solving such as work on inference network architectures [3] and blackboard architectures [14], both of which focus on controling search to produce the best solution possible within a fixed deadline; and agent architectures such as the Phoenix project's agent-based system for on-line planning, scheduling, execution, and monitoring [26].

"Compilation" has various uses in the context of knowledge-based diagnosis. Some of the earliest references are by Chandrasekaran and his students in the domain of nonreal-time medical diagnosis, where the term denotes "compiling knowledge in a form ready to be used for a class of problems of a given type," typically compiling knowledge into structures specialized and tuned for specific types of medical problem solving [5, p. 435]. By contrast, much of the recent work appears related to automated modeling; examples of this type of compilation strategy includes the Rule Set Processor (RSP), a knowledge compilation system contracted by NASA which exploits the advantages of expert systems during the development phase and then compiles the knowledge base into a conventional program for a target embedded microprocessor [15]. The ABE/RT system classified above as a reorganization strategy also includes an executive and model compiler which implements the runtime functions of the ABE/RT languages and translates the resulting model into C++ code frames suitable for prototyping [30]. The MOLTKE system, a nonreal-time expert system for diagnosing CNC machining centers developed at the University of Kaiserslautern [39] is a further example; MOLTKE automatically derives a causal model from technical diagrams of a device and compiles this knowledge into a rule base. The system is interesting because it appears to be a nice synthesis of qualitative reasoning, model-based diagnosis, and pragmatic systems engineering. The common theme of all of the preceding work is the notion of *satisficing* problem solving, i.e., decision methods which seek a *satisfactory*, or best possible solution satisfying given time and resource constraints, rather than an optimal solution.[5]

[5] The term "satisficing" was introduced by H. Simon in a series of classic essays delivered as invited lectures at MIT [43].

2.3 Related Work

After completing the survey of knowledge-based approaches to diagnosis summarized above, we realized that there had been little work on the foundations of recovery and reconfiguration, and virtually none on the problem of integrated FDIR, although there had been some related developments. Poole [35, p. 1310] had noted the generality of the model-based paradigm:

> "Much of the discussion ... has not been specific to diagnosis, but can be applied to any recognition task, where the problem is to determine what is in a system (or a picture) based on observations of the system. For example, one can see [27] as using the idea of consistency-based diagnosis with faults corresponding to plan objects."

There had also been extensions to Reiter's algorithm, such as the work of Ng [32] to extend the algorithm to handle time-varying physical devices.

However, while Reiter's theory of diagnosis captures many of the ideas underlying model-based diagnosis, when we began our work on the topic of diagnosis and recovery in the fall of 1990, there were effectively no published articles that dealt with issues of reconfiguration or recovery. Although the number of researchers in this area is still surprisingly few, the situation has changed within the last few years as questions of the utility of model-based diagnosis have brought these issues to the fore. Work in this area falls into two broad classes: research on repair within the logic- or consistency-based paradigm, such as that of [18,20], and research that attempts to redefine the logic-or consistency-based characterization, such as that of [37]. Since our work falls in the first category, we look most closely at the work of Friedrich and his colleagues. The reader is referred to the last section of Chapter 5 for a thorough discussion of the ideas we summarize here. Friedrich and colleagues [18] define a notion of "therapy" and sketch an algorithm for "the standard therapeutic approach," which can be characterized as a process of interleaving diagnosis and repair to suppress "undesired symptoms." This approach differs from ours in that it eliminates or repairs only those components whose treatment causes the disappearance of the observed symptoms; it assumes that granularity of reconfiguration is precisely that of diagnosis, i.e., the reconfigurable units are the same as the diagnosable units; and it assumes that the level of acceptable system functionality remains constant from diagnosis to reconfiguration. More recent work by Friedrich et al. generalizes the repair algorithm using a temporal framework [20], and Nejdl [31] adds the concepts of belief revision to capture an interleaved diagnosis and repair process.

Provan and Poole's characterization of diagnosis proceeds, like ours, from the idea that diagnostic reasoning must be grounded in issues of utility. Although the general thrust of their arguments for integrating notions of repair into the diagnosistic process is similar to ours, their approach differs in that it focuses on a new characterization of the space of possible diagnoses, based on equivalence classes of diagnoses, in which consistency-based diagnoses constitute one such class. Provan and Poole argue that the notion of use-equivalent class is both more general and more computationally attractive.

It is precisely the dearth of published work on the foundations of recovery and reconfiguration and on integrated FDIR that led us to the work discussed in this report. Our focus has been a formalization of the notion of reconfiguration/recovery in the framework of a general theory of fault diagnosis. Our motivation was to bring to FDIR the clarification and the formal basis for comparing various methods of reconfiguration that Reiter's theory has provided for fault diagnosis. In the following chapter we review Reiter's theory of diagnosis, which provides the context for our subsequent development of an analogous theory of reconfiguration.

Chapter 3

Reiter's Theory of Diagnosis from First Principles

Reiter's formulation of the diagnosis problem [40] can be informally described as follows. Given a description of the design or structure of a physical system and an observation of its behavior which differs from that expected, the goal of diagnosis is to find a set of components whose abnormality explains the discrepancy between the observed and the expected system behavior. The system description is couched in terms of the assumed nonabnormality of its components: e.g., if a light bulb is not abnormal, and has voltage applied, the bulb will be lit. In the simplest realizations of this approach, the system description specifies the behavior of nonabnormal components only; later formulations have augmented the system description with axioms for physical negation, i.e., physically impossible behavior [19], and with explicit "fault models" [44].

3.1 The Formal Characterization

Reiter's formal characterization of diagnosis is stated in terms of two basic theorems, which we reproduce below following four preliminary definitions. For the definitions it is useful to remember that we are taking unions over sets of clauses, yielding conjunctions of first-order sentences. We first characterize a system and its observations.

Definition 1 A **system** is a pair **(sd, comps)** where **sd**, the *system description*, is a set of first-order sentences and **comps**, the *system components*, is a finite set of constants. An *observation*, **obs**, of a system is a finite set of first-order sentences. Thus **(sd, comps, obs)** denotes a system **(sd, comps)** with observation **obs** [40, pp. 59, 62].

Using the predicate **ab** to denote abnormality, Reiter initially characterizes a diagnosis as follows.[1]

Definition 2 A **diagnosis** for **(sd, comps, obs)** is a minimal set $\Delta \subseteq$ **comps** such that $\mathbf{sd} \cup \mathbf{obs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{ab}(c) | c \in \mathbf{comps} - \Delta\}$ is consistent.

Reiter's subsequent characterization of diagnosis exploits the notion of a conflict set first introduced by de Kleer.[2]

Definition 3 A *conflict set* for **(sd, comps, obs)** is a set $\{c_1, \ldots, c_k\} \subseteq$ **comps** such that $\mathbf{sd} \cup \mathbf{obs} \cup \{\neg\mathbf{ab}(c_1), \ldots, \neg\mathbf{ab}(c_k)\}$ is inconsistent. A conflict set for **(sd, comps, obs)** is **minimal** iff no proper subset of it is a conflict set for **(sd, comps, obs)** [40, p. 67].

The final definition characterizes a *hitting set*. Let C be a collection of sets.

Definition 4 A **hitting set** for C is a set $H \subseteq \bigcup_{S \in C} S$ such that $H \cap S \neq \{\ \}$ for each $S \in C$. A hitting set for C is **minimal** iff no proper subset of it is a hitting set for C [40, p. 67].

Given the above definitions, Reiter's principal characterization of diagnoses and the basis for his algorithm are given by the following theorem.

Theorem 1 $\Delta \subseteq$ **comps** is a diagnosis for **(sd, comps, obs)** iff Δ is a minimal hitting set for the collection of conflict sets for **(sd, comps, obs)** [40, p. 67].

Reiter's algorithm for computing diagnoses follows directly from this theorem; the approach calls for computing the minimal hitting sets for an arbitrary collection of sets by generating a hitting set or HS-tree.

Definition 5 Suppose F is a collection of sets. An edge- and node-labeled tree T is an HS-tree for F iff it is a smallest tree with the following properties [40, p. 69]:

[1] Reiter's use of the **ab** predicate derives from McCarthy's use of an abnormality predicate in his formalization of circumscription [40, p. 62].

[2] The original reference appeared in a 1976 MIT AI memo titled "Local Methods for Localizing Faults in Electronic Circuits."

1. Its root is labeled by "$\surd$" if F is empty. Otherwise, its root is labeled by a set in F.

2. If n is a node of T, define $H(n)$ to be the set of edge labels on the path from T from the root node to n. If n is labeled by $\surd$, it has no successor nodes in T. If n is labeled by a set Σ of F, then for each $\sigma \in \Sigma$, n has a successor node in n_σ joined to n by an edge labeled by σ. The label for n_σ is a set $S \in F$ such that $S \cap H(n_\sigma) = \{\ \}$ if such a set S exists. Otherwise, n_σ is labeled by $\surd$.

Reiter notes the following two properties of any HS-tree for a collection of sets. First, if n is a node of the tree labeled by $\surd$, then $H(n)$ is a hitting set for F. Second, each minimal hitting set for F is $H(n)$ for some node n of the tree labeled by $\surd$. Thus, given a set F, the HS-tree given by the preceding definition includes all *minimal* hitting sets for F.[3]

Note that if the collection F of (all) conflict sets is given explicitly, it is reasonable to assume that F can be prescanned and all supersets of sets in F removed. However Reiter's algorithm assumes that F is (only) **implicitly** defined for a given system and observation **(sd, comps, obs)**. As a result, the HS-tree must be pruned during generation. Reiter's strategy assumes that the HS-tree is always generated breadth-first, in left-to-right order. Furthermore, node labels are reused wherever possible; in particular, if node n is labeled by the set $S \in F$ and if n' is a node such that $H(n') \cap S = \{\ \}$, label n' by S.[4]

Definition 6 A **pruning strategy** for an HS-tree T is given by the following three steps [40, p. 72]:

1. If node n is labeled by $\surd$ and node n' is such that $H(n) \subseteq H(n')$, close n', i.e., do not compute a label or any successors for n'.

2. If node n has been generated and node n' is such that $H(n') = H(n)$, close n'.

3. If nodes n and n' have been respectively labeled by sets S and S' of F and if $S' \subset S$, then for each $\alpha \in (S - S')$ remove the redundant

[3] Although not necessarily all hitting sets for F.

[4] The motivation for reusing node labels follows directly from the fact that F is implicit; access to F is in fact a call to an underlying theorem prover which returns a suitable conflict set, where "suitable" means a set S such that $H(n) \cap S = \{\ \}$ if such a set exists, otherwise the theorem prover returns $\surd$. Reusing node labels is one way of minimizing the expense of invoking the theorem prover.

edge from node n labeled by α along with any subtree beneath it and relabel n[5]

The following theorem summarizes the preceding discussion.

Theorem 2 Let F be a collection of sets, and T a pruned HS-tree for F. Then $\{H(n)|n$ is a node of T labeled by $\surd\}$ is the collection of minimal hitting sets for F [40, p. 72].

It should be clear that if F is the collection of conflict sets for **(sd, comps, obs)**, the collection $\{H(n)|\ldots\}$ of minimal hitting sets is precisely the set of all diagnoses for **(sd, comps, obs)**. Note that since pruned HS-trees are generated breadth-first, nodes labeled by $\surd$ at level 1 in the tree correspond to all diagnoses involving a single component, and similarly for diagnoses of increasing cardinality [40, p. 79]. Figure 3.1 is taken from Reiter and shows a pruned HS-tree for the explicit set $F = \{\{2,4,5\},\{1,2,3\},\{1,3,5\},\{2,4,6\},\{2,4\},\{2,3,5\},\{1,6\}\}$ [40, p. 73]; "x" represents a closed branch, "$\surd$" the terminal node of a path corresponding to a minimal hitting set, and "//" a redundant branch which has been pruned.

Reiter's theory is general enough to capture the essence of several other strategies, including those of Davis [7] and de Kleer and Williams [12], assuming certain additional constraints, such as, with respect to de Kleer and Williams, the constraint that all conflict sets are minimal.

3.2 Questions about the Specification of Reiter's Algorithm

Greiner and his colleagues have recently published a research note detailing two problems with Reiter's algorithm [22]. The first difficulty arises out of Reiter's use of nonmimimal conflict sets; in particular, the closing rules fail to take into account the possibility that a branch assumed active by a closing rule might subsequently be pruned, thereby possibly eliminating the path to potential hitting sets, as illustrated in Figure 3.2 [22, p. 82].

The second difficulty relates to Reiter's failure to explicitly specify node relabeling when redundant edges are pruned. As a result, nodes which should remain active may be closed, as in Figure 3.3 which illustrates the

[5] As Greiner et al. point out [22, p. 83], Reiter mentions relabeling in the discussion, but fails to incorporate it into the algorithm.

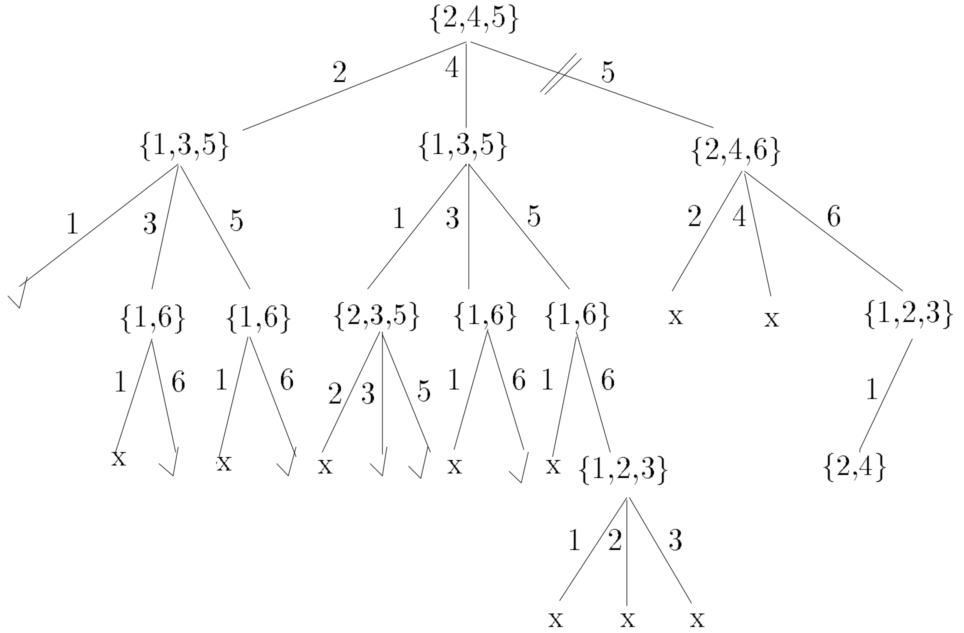

Figure 3.1: Pruned HS-tree

extreme case in which all nodes except the root node would be pruned unless the root node is relabeled when the left branch is pruned [22, p. 83].

Greiner et al. propose a revised version of Reiter's algorithm using directed acyclic graphs, HS-dags, rather than HS-trees, ostensibly as an alternative to reusing node labels. The other difference between their algorithm and Reiter's is their stipulation that the collection F be ordered, thereby yielding a deterministic algorithm. Although Greiner et al. do not discuss the issues, there are clearly tradeoffs. One of the advantages of Reiter's algorithm is its relative simplicity of statement; the algorithm is somewhat more perspicuous than that of Greiner et al. Additionally, Greiner and colleagues fail to mention that their algorithm assumes that F is explicit, whereas Reiter makes precisely the opposite assumption. One way to offset the expense of explicitly generating F is to compile the set prior to its use for diagnosis. A further advantage of this approach is that F can be prescanned for supersets. It seems reasonable to assume that these tradeoffs would be most effectively evaluated in a particular diagnostic context. There have been

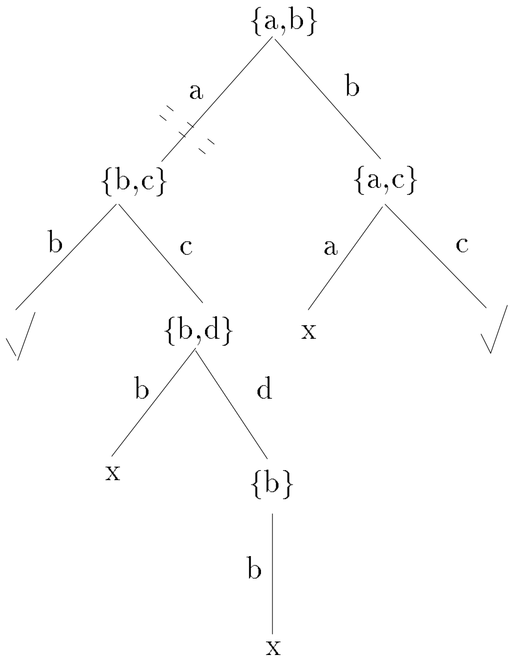

Figure 3.2: Pathological interaction of closing and pruning rules

revisions to the version of Reiter's theory presented in this chapter, most

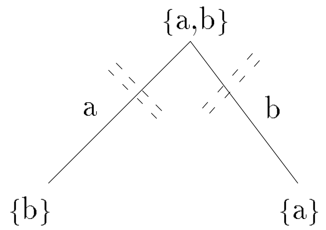

Figure 3.3: Necessity of node relabeling

notably the previously mentioned work of Struss and colleagues [44] on explicit fault models, the work of Friedrich et al. [19] on physical impossibility, and the collaboration of Reiter, de Kleer, and Mackworth [10] resulting in a new characterization of diagnosis. We have chosen to use Reiter's original formulation as the basis for our work because the newer characterization has only recently become available and is not yet as widely disseminated. This completes our review of Reiter's theory of diagnosis; we now have the necessary background to consider extensions to the basic theory. The extension we have in mind, of course, is a complementary formalization of reconfiguration, the subject of the following chapter.

Chapter 4

A Theory of Reconfiguration from First Principles

We are concerned with the problem of reconfiguration in the context of systems designed for survivability, and therefore provided with some degree of fault tolerance and redundancy. In this type of system, components may be capable of operating in different modes (e.g., standard or degraded mode), may be switched off or bypassed, and may have standby spares or other forms of redundancy. In addition to being able to reconfigure components, we may also be willing to accept certain behaviors other than that considered truly correct. For example, a system may be required to withstand two component failures with no loss of capability, but may be allowed to degrade to a safe mode on the third failure. We make no particular assumptions about redundancy or degradation of performance; our theory is general enough to accommodate the range of possibilities suggested here. Thus a system description for any type of redundant and/or degradable system can be given in terms of the reconfiguration of its components; e.g., if system x is reconfigured, then receiver 1 is in the circuit, otherwise receiver 2, or if system y is reconfigured then the transmitter operates at half power otherwise full power.

By analogy with Reiter's formulation of diagnosis, the problem of reconfiguration can be posed as follows. Given a diagnosis, i.e., a set of components assumed abnormal, find a set of components whose reconfiguration yields an acceptable behavior. In particular, given a system and diagnosis, find a reconfiguration *relative to the diagnosis* Δ, i.e., a set of components whose reconfiguration yields an acceptable behavior. Note that any such reconfiguration assumes the abnormality of the components in Δ.

A simple example should help clarify these notions. Consider the problem of diagnosing and repairing a flat on a bike equipped with a single spare tire. To simplify the statement of the problem, we use a typed logic. *Wheel* and *tire* are uninterpreted types, *front* and *back* are constants of type *wheel*, x is a variable of the same type, and a, b and *spare* are constants of type *tire*. The function *on* has signature *wheel* $\rightarrow$ *tire* and indicates which tire is on which wheel, *good* and *rcfg* are predicates on wheels, and *ab* is a predicate on tires. Intuitively, *ab* indicates whether or not a tire is serviceable, $rcfg(x)$ indicates whether the spare is to be mounted on wheel x, and *good* indicates whether or not a wheel has a serviceable tire. In this and subsequent discussion, we make the simplifying assumption that components used in reconfiguration are not abnormal; in this case, the spare tire is assumed serviceable. The system description is as follows.

$$\neg ab(on(x)) \supset good(x)$$
$$rcfg(x) \supset on(x) = spare$$
$$\neg rcfg(front) \supset on(front) = a$$
$$\neg rcfg(back) \supset on(back) = b$$
$$\neg rcfg(front) \vee \neg rcfg(back)$$
$$\neg rcfg(front) \wedge \neg rcfg(back)$$

The last of these axioms indicates the initial configuration—i.e., neither wheel is reconfigured. Suppose we notice that our back tire is rapidly loosing air, i.e.,

$$\neg good(back).$$

From the model, we discover there is a single diagnosis $\{b\}$, i.e., $ab(b)$ is consistent with the model and the observation. We now add

$$ab(b) \wedge \neg ab(a) \wedge \neg ab(spare)$$

to the system description, withdraw the initial configuration

$$\neg rcfg(front) \wedge \neg rcfg(back),$$

establish the requirement

$$good(front) \wedge good(back),$$

and seek a reconfiguration that is consistent with this requirement and the system description. Clearly $\{back\}$ does the job; we should put the spare on the back wheel.

Note that there are two phases to this approach: first we fix the configuration and seek diagnoses, then we fix a diagnosis and seek a reconfiguration. In general, there will be several diagnoses and we will probably seek a reconfiguration for each before committing to a final choice.

Although very simple, this example illustrates an important point: the concept of reconfiguration can be decoupled from the mechanisms for achieving it. Of course, it is also possible to equate the concept and mechanism of reconfiguration, as illustrated in the example in Section 5.1 of Chapter 5, where the reconfiguration predicate is applied to switches in an electrical circuit.

We are now ready to discuss the formal development of the analogy between diagnosis and reconfiguration. To begin, we present the theory of reconfiguration underlying our claim that the problem of reconfiguration can be viewed as a close analogue of the problem of diagnosis, thereby establishing a formal basis for reconfiguration which closely parallels that for diagnosis. In fact, it turns out that the generality of Reiter's theory renders it equally applicable to reconfiguration and Reiter's proofs [40] go through virtually unchanged.

4.1 An Intuitive Characterization of Reconfiguration

Let $\mathbf{comps}'$ be the union of **comps** and any additional components such as standby spares or auxiliaries available for recovery; $\mathbf{sd}'$ be the union of **sd** and any additional configuration statements or correct behavior axioms for the spare components; and let the requirements, **reqs**, be a finite set of first-order sentences specifying desired or acceptable behavior for the reconfigured system. The predicate **rcfg** denotes "reconfigured." Thus for component c, $\mathbf{ab}(c)$ denotes that c is not behaving normally and $\neg\mathbf{rcfg}(c)$ denotes that c is not reconfigured. We begin with a basic definition of reconfiguration relative to a diagnosis. In the definitions and proofs which follow, it is useful once again to point out that we are taking unions over sets of clauses, yielding conjunctions of first-order sentences. To facilitate comparison between our proofs and Reiter's, we have retained his use of the term "proposition," although we prefer to think of Propositions 1-4 as lemmas, trivial lemmas in the case of Propositions 1 and 2, and to view Proposition 5 as a corollary of Proposition 4.

Definition 7 A **reconfiguration** for **(sd′, comps′, reqs)** relative to Δ is a minimal set $\Re_\Delta \subseteq \mathbf{comps}'$ such that

$$\mathbf{sd}' \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \mathbf{reqs} \cup \{\mathbf{rcfg}(c) | c \in \Re_\Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\}$$

is consistent.

Definition 7 characterizes a reconfiguration relative to a diagnosis as the smallest set of components such that the assumption that these components are reconfigured and that all other components are not reconfigured is consistent with the diagnosis, the augmented system description, and the requirements. Note that we do not need to add the set $\{\neg\mathbf{ab}(c) | c \in \mathbf{comps} - \Delta\}$ to the union in this definition because the corresponding definition of diagnosis [40, p. 63, Def. 2.4] specifies the consistency of

$$\mathbf{sd} \cup \mathbf{obs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{ab}(c) | c \in \mathbf{comps} - \Delta\}.$$

The approach to reconfiguration suggested here provides insight, but ultimately not much practicality, as we will see in due course. The next two propositions follow straightforwardly from Definition 7.

Proposition 1 A reconfiguration exists for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs})$ relative to Δ IFF

$$\mathbf{sd}' \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \mathbf{reqs}$$

is consistent.

Note that Δ may be the empty set.

Proposition 2 $\{\ \}$ is the unique reconfiguration for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs})$ relative to Δ IFF

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}'\}$$

is consistent, i.e., if the requirements are consistent with the system behavior in the presence of the faults indicated by Δ.

Proposition 3 characterizes the relation between reconfigured and unreconfigured components; the latter can be said to "logically determine" the former.

Proposition 3 If $\Re_\Delta$ is a reconfiguration for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs})$ relative to Δ, then for each $c_i \in \Re_\Delta$,

$$\mathbf{sd'} \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps'} - \Re_\Delta\} \models \mathbf{rcfg}(c_i).$$

PROOF If $\Re_\Delta$ is the empty set, then the proposition holds vacuously. Let $\Re_\Delta = \{c_1, \ldots, c_k\}$ and assume that the proposition is false, i.e., that

$$\mathbf{sd'} \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps'} - \Re_\Delta\} \\ \cup \{\neg\mathbf{rcfg}(c_1) \vee \ldots \vee \neg\mathbf{rcfg}(c_k)\}$$

is consistent. Clearly,

$$\{\neg\mathbf{rcfg}(c_1) \vee \ldots \vee \neg\mathbf{rcfg}(c_k)\}$$

is logically equivalent to

$$\bigvee [X_1(c_1) \wedge \ldots \wedge X_k(c_k)],$$

where

$$\exists X_j,\ 1 \leq j \leq k,\ X_j = \neg\mathbf{rcfg}(c_j)$$

and $\forall i \neq j, 1 \leq i \leq k$ the term X_i is either $\mathbf{rcfg}(c_i)$ or $\neg\mathbf{rcfg}(c_i)$, i.e., there is at least one conjunct $\neg\mathbf{rcfg}(c_j)$ in each of the disjunctions. It follows that

$$\mathbf{sd'} \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps'} - \Re_\Delta\} \\ \cup \bigvee [X_1(c_1) \wedge \ldots \wedge X_k(c_k)]$$

is consistent. Hence for some choice of conjunction over $X_1 \ldots X_k$,

$$\mathbf{sd'} \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps'} - \Re_\Delta\} \\ \cup \{X_1(c_1) \wedge \ldots \wedge X_k(c_k)\}$$

is consistent. But this means that $\Re_\Delta$ has a strict subset $\Re'_\Delta$ such that

$$\mathbf{sd'} \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps'} - \Re'_\Delta\} \\ \cup \{\mathbf{rcfg}(c) | c \in \Re'_\Delta\}$$

is consistent, contradicting the hypothesis that $\Re_\Delta$ is a reconfiguration for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs})$ relative to Δ. $\Box$

This is Reiter's proof essentially unchanged except for notation and the obvious substitutions to accommodate reconfiguration. Like the original, it seems unnecessarily opaque; why bother with the disjunction over all c_i, i.e., why not take the case of a single $c_i \in \Re_\Delta$ as follows?

ALTERNATE PROOF Suppose that the proposition is false and $\exists c_i \in \Re_\Delta$ such that

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\} \not\models \mathbf{rcfg}(c_i),$$

i.e.,

$$\begin{aligned}\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\} \\ \cup\{\neg\mathbf{rcfg}(c_i)\}\end{aligned}$$

is consistent. But this means that $\Re_\Delta$ has a strict subset $\Re_\Delta^{'}$ such that

$$\begin{aligned}\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta^{'}\} \\ \cup\{\mathbf{rcfg}(c) | c \in \Re_\Delta^{'}\}\end{aligned}$$

is consistent, contradicting the hypothesis that $\Re_\Delta$ is a reconfiguration for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs})$ relative to Δ. $\square$
The addition of the last clause is justified by the fact that

$$\{\mathbf{rcfg}(c) | c \in \Re_\Delta^{'}\} \subset \{\mathbf{rcfg}(c) | c \in \Re_\Delta\}$$

and $\Re_\Delta$ is a reconfiguration (*cf.* Definition 7). From the definition of reconfiguration (Definition 7) and Proposition 3, we can establish the following.

Proposition 4 $\Re_\Delta \subseteq \mathbf{comps}'$ is a reconfiguration for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs})$ relative to Δ IFF $\Re_\Delta$ is a minimal set such that

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\}$$

is consistent.

PROOF $\Rightarrow$. The result follows straightforwardly from Proposition 3 and the definition of reconfiguration. From Definition 7, we have that

$$\begin{aligned}\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\} \\ \cup\{\mathbf{rcfg}(c) | c \in \Re_\Delta\}\end{aligned}$$

is consistent, and thus that

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta\}$$

is consistent. Note that $\Re_\Delta$ is defined to be minimal with respect to the first property given above, but might not be minimal with respect to the second, hence we must prove $\Re_\Delta$ minimal in the context of

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta\}.$$

Since by Proposition 3, $\forall c_i \in \Re_\Delta$,

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta\} \\ \cup\{\neg\mathbf{rcfg}(c_i)\}$$

is inconsistent, it follows that $\Re_\Delta$ is a minimal set with the desired property.

$\Leftarrow$. We must show that $\Re_\Delta$ is a minimal set satisfying the definition of a reconfiguration (Definition 7). Given that $\Re_\Delta$ is minimal, $\forall c_i \in \Re_\Delta$,

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta\} \cup \{\neg\mathbf{rcfg}(c_i)\}$$

is inconsistent. But, by hypothesis,

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta\}$$

is consistent and thus

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta\} \cup \{\mathbf{rcfg}(\mathbf{c})|c \in \Re_\Delta\}$$

is consistent. Furthermore, $\Re_\Delta$ is minimal, since otherwise there would be a set $\Re_\Delta' \subset \Re_\Delta$ such that

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|c \in \mathbf{comps}' - \Re_\Delta'\}$$

is consistent, contradicting the hypothesis of this proposition. □

By proposition 4, $\Re_\Delta$ is a reconfiguration IFF $\{\neg\mathbf{rcfg}(c)|\mathbf{comps}' - \Re_\Delta\}$ is consistent with

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\},$$

i.e., just in case

$$\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c)|c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c)|\mathbf{comps}' - \Re_\Delta\}$$

is not inconsistent.

At this point it is useful to return to the question of practicality. We have captured the intuition that a reconfiguration is a conjecture that recovery can be achieved by reconfiguring only certain components, but we have not provided the basis for an efficient mechanism for computing all reconfigurations. Given the proof of Proposition 4, the reconfiguration analogue to Reiter's Proposition 3.4, we could systematically generate subsets $\Re_\Delta$ of **comps**, starting with those of minimal cardinality, and test the consistency of

$$\mathbf{sd}' \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \mathbf{reqs} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\}.$$

For systems with numerous components, this approach is clearly unacceptably inefficient. In the next section we develop a more effective basis for reconfiguration.

4.2 Characterizing the Computation of a Reconfiguration

Following Reiter, we exploit the notion of conflict set and hitting set to arrive at a more effective computational basis.[1]

Definition 8 A **conflict set** for **($\mathbf{sd}'$, $\mathbf{comps}'$, reqs, Δ)** is a set $\{c_1, \ldots, c_k\} \subseteq \mathbf{comps}'$ such that $\mathbf{sd}' \cup \mathbf{reqs} \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{rcfg}(c_1) \wedge \ldots \wedge \neg\mathbf{rcfg}(c_k)\}$ is inconsistent. A conflict set for **($\mathbf{sd}'$, $\mathbf{comps}'$, reqs, Δ)** is **minimal** IFF no proper subset of it is a conflict set for **($\mathbf{sd}'$, $\mathbf{comps}'$, reqs, Δ)**.

Although this definition of conflict set appears identical to Reiter's with the exception of the change in names and the introduction of the context of diagnosis, the appropriate interpretation of the definition is not obvious. In diagnosis, a conflict set reflects the fact that if all of the components named as elements of the conflict set work, i.e., are not abnormal ($\neg$**ab**), the observation and the system description are inconsistent. In reconfiguration, the inconsistency arises if all of the components in the conflict set are unreconfigured, i.e., are not reconfigured ($\neg$**rcfg**). Hence Proposition 4 can be reformulated as

[1] The notion of a conflict set was originally proposed by de Kleer and later formalized by Reiter [40, p. 67]. The notion of a hitting set also appears in Reiter [40, p. 67].

Proposition 5 $\Re_\Delta \subseteq \mathbf{comps'}$ is a reconfiguration for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs})$ relative to Δ IFF $\Re_\Delta$ is a minimal set such that $\mathbf{comps'} - \Re_\Delta$ is not a conflict set.

Given a collection of sets, in this case a collection of conflict sets, the notion of a hitting set is defined as follows.

Definition 9 A **hitting set** for a collection of sets C is a set $H \subseteq \bigcup_{S \in C} S$ such that $H \cap S \neq \{\ \}$ for each $S \in C$. A hitting set for C is **minimal** IFF no proper subset of it is a hitting set for C [40, p. 67].

for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs}, \Delta)$.

We can now characterize the computation of a reconfiguration as follows.

Theorem 3 $\Re_\Delta \subseteq \mathbf{comps'}$ is a reconfiguration for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs})$ relative to Δ IFF $\Re_\Delta$ is a minimal hitting set for a collection of conflict sets containing at least the minimal conflict sets for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs}, \Delta)$.[2]

PROOF $\Rightarrow$. From Proposition 4.2, we have that $\mathbf{comps'} - \Re_\Delta$ is not a conflict set for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs}, \Delta)$. Therefore every conflict set must include an element of $\Re_\Delta$, which means that $\Re_\Delta$ is a hitting set for the collection of conflict sets for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs}, \Delta)$. Furthermore, since $\Re_\Delta$ is a minimal set such that $\mathbf{comps'} - \Re_\Delta$ is not a conflict set,

$$\forall c \in \Re_\Delta, \{c\} \cup \{\mathbf{comps'} - \Re_\Delta\}$$

is a conflict set. Hence $\Re_\Delta$ is a minimal hitting set for the conflict sets for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs}, \Delta)$.

$\Leftarrow$. Using Proposition 4.2, we prove that $\Re_\Delta$ is a reconfiguration for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs})$ relative to Δ by proving that

1. $\mathbf{comps'} - \Re_\Delta$ is not a conflict set for $(\mathbf{sd'}, \mathbf{comps'}, \mathbf{reqs}, \Delta)$.

2. $\Re_\Delta$ is minimal with respect to property 1.

[2] Reiter's statement of the corresponding theorem for diagnosis is vague, specifying only that Δ be a minimal hitting set "for the collection of conflict sets for (**sd**, **components**, **obs**)." In fact, the collection must include at least the minimal conflict sets, or the diagnosis may be erroneous. The same caveat applies to reconfiguration; the relevant collection must include at least the minimal conflict sets.

Proof of 1. Assume the contrary, i.e., $\mathbf{comps}' - \Re_\Delta$ is a conflict set for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs}, \Delta)$. But then $\Re_\Delta \cap (\mathbf{comps}' - \Re_\Delta) = \{\ \}$, contradicting the hypothesis that $\Re_\Delta$ is a hitting set for all conflict sets.

Proof of 2. We prove that $\Re_\Delta$ is minimal with respect to property 1 by showing that $\forall c \in \Re_\Delta$, $\{c\} \cup \mathbf{comps}' - \Re_\Delta$ is a conflict set for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs}, \Delta)$. Since $\Re_\Delta$ is a minimal hitting set for the collection of conflict sets for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs}, \Delta)$, it follows that $\forall c \in \Re_\Delta$, $\exists X \subseteq \mathbf{comps}' - \Re_\Delta$ such that $\{c\} \cup X$ is a conflict set for $(\mathbf{sd}', \mathbf{comps}', \mathbf{reqs}, \Delta)$. If not, $\exists c' \in \Re_\Delta$ such that $\{c\} \cup \{c'\} \cup X$ is a conflict set, in which case $\Re_\Delta - \{c\}$ is a smaller hitting set than $\Re_\Delta$, contradicting the hypothesis that $\Re_\Delta$ is a minimal hitting set. Furthermore, if $\{c\} \cup X$ is a conflict set, $\{c\} \cup \mathbf{comps}' - \Re_\Delta$ is a conflict set. Hence $\Re_\Delta$ is a minimal set such that $\mathbf{comps}' - \Re_\Delta$ is not a conflict set. $\Box$

Having provided a formal basis for viewing the problem of reconfiguration as an analogue of that of diagnosis, we set aside formal considerations and turn to a series of examples illustrating the application of these ideas.

Chapter 5

A Set of Related Examples

To help the reader understand these examples as illustrations of the theory of reconfiguration presented in Chapter 4, we suggest the following intuition. Diagnosis and reconfiguration can both be thought of as strategies for generating alternative designs or views of the system. If diagnosis of a system yields three candidate diagnoses and two possible reconfigurations, we view this outcome as three *diagnosis designs* and two *reconfiguration designs*. The reconfiguration mechanisms mentioned previously can then be interpretated as mechanisms for switching between or selecting among the reconfiguration designs. This is an important point, because we do not want to limit ourselves to cases in which the granularity of the reconfiguration is identical to that of diagnosis. Nor do we want to limit ourselves to cases where REQS simply specifies the original functionality. In other words, the *reconfiguration design* problem is in some sense much richer than the *diagnosis design* problem; the reconfiguration design problem has several additional parameters to work with, including (potentially major) changes in functionality and structure, and this is what makes the problem of FDIR and, in particular, integrated FDIR interesting. Even in examples where the granularity of reconfiguration and diagnosis is identical, it is important to realize that unlike the components used in diagnosis, reconfiguration mechanisms represent 'meta-level' elements rather than physical elements. Of course, it is certainly possible to axiomatize the behavior of components that actually perform reconfiguration, such as switches and valves; there is nothing to preclude identifying meta-level components with physical components, as long as we recognize that this is simply one instantiation of the basic approach.

5.1 A Standard Example

Our first set of examples is based on minor variations of a standard example (see, for example, [19, p. 332]) consisting of a battery and a series of bulbs connected in parallel as shown in Figure 5.1. We have added three reconfiguration switches (r) normally set so that the standby spares and auxiliary bulb are wired into the circuit as indicated.

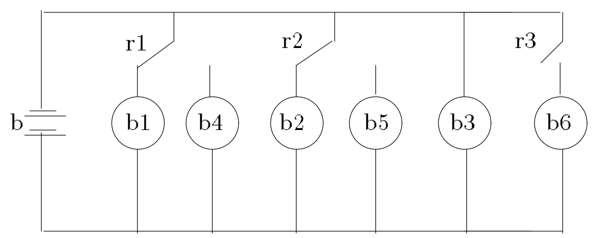

Figure 5.1: Simple circuit with auxiliary bulb and two standby spares

We specify the **comps** of the system in Figure 5.1 as $\{b, b1, b2, b3, b4, b5, b6, r1, r2, r3\}$ and, using the abnormality predicate **ab** and the reconfiguration predicate **rcfg**, we specify the sentences that constitute the **sd** as shown below; the first six sentences axiomatize the correct behavior of the components and the last three describe the physical configuration. Variables are denoted by capital letters and are implicitly universally quantified. We assume that wires always behave correctly.

$$bulb(X) \wedge \neg ab(X) \wedge powered(X) \supset lit(X) \tag{1}$$
$$bulb(X) \wedge \neg ab(X) \wedge \neg powered(X) \supset \neg lit(X) \tag{2}$$
$$bulb(X) \wedge \neg ab(X) \wedge lit(X) \supset powered(X) \tag{3}$$
$$bulb(X) \wedge \neg ab(X) \wedge \neg lit(X) \supset \neg powered(X) \tag{4}$$
$$battery(X) \wedge \neg ab(X) \supset powered(X) \tag{5}$$
$$wired(X,Y) \supset powered(X) \equiv powered(Y) \tag{6}$$
$$battery(b) \wedge bulb(b1) \wedge \ldots \wedge bulb(b6) \tag{7}$$
$$\neg rcfg(r1) \wedge \neg rcfg(r2) \wedge \neg rcfg(r3) \tag{8}$$
$$\begin{gathered} wired(b,b1) \equiv \neg rcfg(r1) \wedge wired(b,b2) \equiv \neg rcfg(r2) \wedge wired(b,b3) \\ \wedge wired(b,b4) \equiv rcfg(r1) \wedge wired(b,b5) \equiv rcfg(r2) \wedge wired(b,b6) \equiv rcfg(r3) \end{gathered} \tag{9}$$

The expected behavior is that bulbs $b1, b2, b3$ are lit. The observation

$$\neg lit(b1) \wedge \neg lit(b2) \wedge lit(b3) \quad (10)$$

yields the following set of conflict sets: $\{\{b, b1\}, \{b, b2\}, \{b1, b3\}, \{b2, b3\}\}$. To illustrate the derivation of conflict sets, we specify the conflicts used in the diagnosis in this first example only; bracketed numbers refer to the corresponding sentences in the **sd**.

$$\begin{array}{ll} \{b, b1\}: & powered(b) \quad [7,5] \\ & \neg powered(b1) \quad [7,10,4] \\ & T \equiv F \quad [6] \end{array}$$

$$\begin{array}{ll} \{b1, b3\}: & \neg powered(b1) \quad [7,10,4] \\ & powered(b3) \quad [7,10,3] \\ & powered(b) \equiv powered(b1) \equiv F \quad [9,6] \\ & powered(b) \equiv powered(b3) \equiv T \quad [9,6] \end{array}$$

The derivations for $\{b, b2\}$ and $\{b2, b3\}$ are analogous, with the obvious substitutions of $b2$ for $b1$. There are two hitting sets for this collection of conflicts, i.e., two candidate diagnoses: $\{b, b3\}$ and $\{b1, b2\}$. This type of example is typically used to illustrate the necessity of augmenting the correct behavior model traditionally assumed in model-based diagnosis with some specification of incorrect behavior, e.g., fault models or physical impossibility axioms. This aspect of the example is irrelevant to our discussion, and we ignore the absurd diagnosis $\{b, b3\}$.

Suppose that the system requirements under reconfiguration, **reqs**, are somewhat weaker than the original functionality: at least two bulbs should be lit, i.e.,

$$\exists X, Y\ (lit(X) \wedge lit(Y)) \quad (11)$$

and the candidate diagnosis is $\{b1,\ b2\}$. Reconfiguration of this system involves no new components, i.e., **comps** = **comps**′ and the modified system description **sd**′ includes the additional sentence: $\mathbf{ab}(b1) \wedge \mathbf{ab}(b2)$. This gives the set of conflict sets $\{\{b4, b5, b6\}\}$ and the candidate reconfigurations $\Re_\Delta = \{b4\} \vee \{b5\} \vee \{b6\}$. In other words, assuming $b3$ lit and three spare bulbs, there are three ways to reconfigure the system satisfying the given requirements. Clearly if the reconfiguration requirements specified the original functionality, i.e.,

$$\exists X, Y, Z\ (lit(X) \wedge lit(Y) \wedge lit(Z)), \quad (12)$$

then the set of conflict sets would be $\{\{b4, b5, b6\}, \{b4, b5\}, \{b5, b6\}, \{b4, b6\}\}$ and the candidate reconfigurations $\Re_\Delta = \{b4, b5\} \vee \{b5, b6\} \vee \{b4, b6\}$.

We can use this example to illustrate two further points. First, suppose that we have no information about which bulbs are lit; we know only that two bulbs have failed, and we require that at least two bulbs be lit. The candidate diagnoses are: $\{b1, b2\}, \{b1, b3\}, \{b2, b3\}$. However, a single reconfiguration, namely $r3$, satisfies **reqs** and there is no need to further discriminate the diagnoses.

Our second point concerns remarks made in the introductory paragraphs of this chapter, where we emphasized that the granularity of reconfiguration need not be identical to that of diagnosis and, more importantly, that reconfiguration switches are meta-level objects which allow us to decouple the concept of reconfiguration from the mechanisms which implement it. A minor variation on our onging example illustrates this point nicely. Suppose that instead of standby sparing for the bulbs, we provide a single flashing yellow bulb as a warning indicator. If at least two of the three bulbs are diagnosed as faulty, the only acceptable reconfiguration is to light the yellow warning bulb. Conceptually, this version of the circuit has two switches: one breaks the connection between the battery and bulbs $b1$, $b2$, $b3$, the other connects the warning indicator $b4$. The circuit and its specification appear below. As before, we assume that wires always behave correctly.

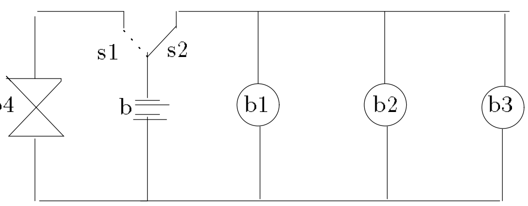

Figure 5.2: Standard circuit with SPDT switch and warning indicator

The **comps** of the system in Figure 5.2 are $\{b$, $b1$, $b2$, $b3$, $b4$, $s1$, $s2\}$. Note that switches $s1$ and $s2$ could be implemented and modeled as a single element, such as a single-pole double-throw (SPDT) switch; we have opted for two separate switches because the distinction between concept and mechanism is more explicit.

$$bulb(X) \wedge \neg ab(X) \wedge powered(X) \supset lit(X) \qquad (1)$$
$$bulb(X) \wedge \neg ab(X) \wedge \neg powered(X) \supset \neg lit(X) \qquad (2)$$
$$bulb(X) \wedge \neg ab(X) \wedge lit(X) \supset powered(X) \qquad (3)$$
$$bulb(X) \wedge \neg ab(X) \wedge \neg lit(X) \supset \neg powered(X) \qquad (4)$$
$$battery(X) \wedge \neg ab(X) \supset powered(X) \qquad (5)$$
$$wired(X, Y) \supset powered(X) \equiv powered(Y) \qquad (6)$$
$$battery(b) \wedge bulb(b1) \wedge \ldots \wedge bulb(b4) \qquad (7)$$
$$\neg rcfg(s1) \wedge \neg rcfg(s2) \qquad (8)$$
$$\neg rcfg(s1) \wedge \neg rcfg(s2) \qquad (9)$$
$$\begin{array}{c} wired(b, b1) \equiv wired(b, b2) \equiv wired(b, b3) \equiv \neg rcfg(s2) \\ \wedge wired(b, b4) \equiv rcfg(s1) \equiv rcfg(s2) \end{array} \qquad (10)$$

Given the observation that at least two bulbs are not lit, i.e.,

$$\exists X, Y\ (\neg lit(X) \wedge \neg lit(Y)), \qquad (11)$$

the diagnosis is moot and the reconfiguration trivial; regardless of which two bulbs are faulty, there is exactly one acceptable recovery, namely powering on the warning light by resetting switches $s1$ and $s2$. There is of course a third possibility: all three bulbs are out because the battery is faulty. Interestingly, the one acceptable reconfiguration also serves to confirm or deny this possibility. Thus this admittedly simple example illustrates the role of reconfiguration in reducing the cost of discriminating among candidate diagnoses, as well as the important distinction between the concept of reconfiguration and the mechanisms for achieving it. Note that although the original system functionality is considerably different under reconfiguration, the granularity of reconfiguration is basically that of diagnosis in this example.

In the next section we turn to an example presented in [18] which serves as a further illustration of our approach as well as a vehicle for comparing Friedrich's therapeutic approach with our notion of recovery.

5.2 A Less Familiar Example: Therapy versus Reconfiguration

Figure 5.3 is a modified version of an electrical circuit which appears in [18, p. 71]. As given in [18], the example consists of a power source, bulb, resistance, and four electrical devices. The power source, bulb, and wires are assumed to operate correctly; the electrical devices either behave normally or produce a short circuit between their respective inputs and outputs. We have augmented the example by introducing three reconfiguration switches and standby spares for three of the four electrical devices. The symbols a, b, c, d in Figure 5.3 are labels identifying locations in the circuit; they are not components.

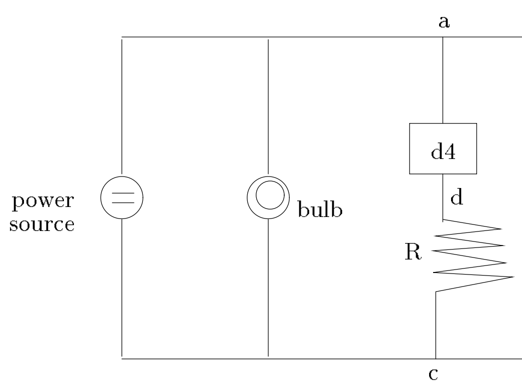

Figure 5.3: An example from the standard therapeutic approach

As reflected in Figure 5.3, the **comps** of the system are $\{ps,\ b,\ R,\ d1,\ d1',\ d2,\ d2',\ d3,\ d3',\ d4,\ r1,\ r2,\ r3\}$. The specification of Friedrich et al. for the original system is written in propositional Horn clause logic and the problem is posed as a propositional Horn clause abduction problem [18, 35, 38]. Briefly, given a logical description of a system, such as a set of propositional Horn clauses, and a set of observations, abductive diagnostic reasoning attempts to find one or more minimal sets of individual hypotheses or diagnoses which logically imply the observations. As in the previous examples, we specify the circuit in first order logic and frame the problem in terms of the consistency of a set of first order formulae. Let $voltage(x, y)$ denote the voltage between point x and point y.

$$power\_source(ps) \wedge bulb(b) \wedge resistance(R) \wedge electrical\_device(d1) \wedge \ldots \wedge electrical\_device(d4) \wedge switch(r1) \wedge \ldots \wedge switch(r3) \tag{1}$$

$$\neg ab(ps) \wedge \neg ab(b) \wedge \neg ab(R) \tag{2}$$

$$\neg rcfg(r1) \wedge \neg rcfg(r2) \wedge \neg rcfg(r3) \tag{3}$$

$$electrical\_device(d1) \wedge \neg ab(d1) \supset voltage(a,b) > 0 \tag{4}$$

$$electrical\_device(d1') \wedge \neg ab(d1') \wedge rcfg(r1) \supset voltage(a,b) > 0 \tag{5}$$

$$electrical\_device(d2) \wedge \neg ab(d2) \supset voltage(a,b) > 0 \tag{6}$$

$$electrical\_device(d2') \wedge \neg ab(d2') \wedge rcfg(r2) \supset voltage(a,b) > 0 \tag{7}$$

$$electrical\_device(d3) \wedge \neg ab(d3) \supset voltage(b,c) > 0 \tag{8}$$

$$electrical\_device(d3') \wedge \neg ab(d3') \wedge rcfg(r3) \supset voltage(b,c) > 0 \tag{9}$$

$$electrical\_device(d4) \wedge \neg ab(d4) \supset voltage(a,d) > 0 \tag{10}$$

$$voltage(a,b) > 0 \vee voltage(b,c) > 0 \supset voltage(a,c) > 0 \tag{11}$$

$$voltage(a,d) > 0 \vee voltage(d,c) > 0 \supset voltage(a,c) > 0 \tag{12}$$

$$voltage(a,b) = 0 \wedge voltage(b,c) = 0 \supset voltage(a,c) = 0 \tag{13}$$

$$voltage(a,c) > 0 \supset lit(b) \tag{14}$$

$$voltage(a,c) = 0 \supset \neg lit(b) \tag{15}$$

Given the observation that the bulb is not lit, i.e.,

$$\neg lit(b), \tag{16}$$

there are two diagnoses for this system: $\{d1, d3\}$ and $\{d2, d3\}$. The algorithm for the standard therapeutic approach [18, p. 75], which interleaves diagnosis and therapy, can be summarized as given below. We have restated the algorithm to avoid introducing new vocabulary, but have otherwise tried to faithfully reproduce the standard therapeutic approach. Let a "potential cause" refer to a (potentially faulty) component, and let C denote the set of all possible causes, i.e., the set of all possibly faulty components. A diagnosis is thus a subset of C.

1. If there are potential causes remaining, i.e., C is nonempty, generate a diagnosis.

2. Check the validity of each individual cause in the diagnosis against the "real world," i.e., verify that the component is actually misbehaving.

3. If the validity of one or more of the causes can not be established, remove the invalid cause(s) from the set C of possible causes and go to step 1.

4. If all causes of the given diagnosis have been validated, perform a *treatment*: repair or eliminate the actual components corresponding to a nonempty subset d' of the diagnosis set; selection of the subset is based on criteria such as cost, feasibility, etc.

5. If the malfunction(s) is(are) still observed, remove the subset d' corresponding to the components repaired or eliminated in step 4 from the set C of possible causes and go to step 1, else stop.

By way of illustration, a possible therapy scenario suggested in [18, p. 77] would play as follows.[1] Let $\{d1, d3\}$ be the first diagnosis generated. This diagnosis is verified (step 2 above) and the component $d1$ is selected for treatment. After eliminating this component, the malfunction persists, i.e., the bulb remains unlit. At this point, rather than treat the component corresponding to the one remaining cause of the diagnosis, $d3$, the algorithm specifies that we generate a new diagnosis, $\{d2, d3\}$. The diagnosis is again verified and this time component $d3$ is selected for treatment. The therapy is successful and the observation invalidated, i.e., the light is now on. The successful therapy involved two components: $d1$ and $d3$.

There are several things to note about the standard therapeutic approach. First, this approach eliminates or repairs only those components whose treatment entails the disappearance of the observed symptoms; there is apparently no attempt to reconcile a given treatment with the specified functionality of the system. Second, given the strategy of interleaving diagnosis and therapy, there is no way to identify minimal or parsimonious therapies. Third, a new diagnosis is generated before the causes of the previous diagnosis are exhausted; in the scenario above, we generated a new diagnosis after treating $d1$ rather than treating $d3$, which in this case would have alleviated the symptom (bulb unlit). If the cost of generating diagnoses is high, as one would suspect, this strategy may be unnecessarily expensive, especially in the context of therapy, which specifies relief of symptoms rather than recovery of specified functionality.

Our approach to FDIR is somewhat more flexible; we have the option of either interleaving diagnosis and recovery, or generating a set of diagnoses followed by a corresponding set of reconfigurations. In the latter case, we may potentially simplify the reconfiguration step. For example, if diagnoses overlap, as in this example, we can reflect this fact in the $\mathbf{sd}'$, thereby

[1] We think there may be problems with the therapy algorithm and scenario as stated in [18], and cite their example merely to illustrate the approach.

reducing the number of diagnoses considered. The $\mathbf{sd}'$ for reconfiguration of the circuit in Figure 5.3 would thus include the formula:

$$ab(d3) \wedge (ab(d1) \vee ab(d2)). \qquad (17)$$

On the other hand, if we choose to interleave diagnosis and reconfiguration and opt first to reconfigure relative to the diagnosis $\{d1,\ d3\}$, we would generate a viable recovery, namely reconfigure components $d1$ and $d3$, before generating the second diagnosis. In applications where one recovery option is adequate or even preferred, this is potentially a significant saving. A further difference between our approach and that proposed by Friedrich et al. is that we have chosen to follow Reiter's strategy of working from first principles, whereas Friedrich et al. assume feedback from the real world. Our assumption has been that in application areas such as air and space missions, it is unrealistic to assume that feedback is always available.

The examples cited in this chapter have illustrated our approach to FDIR and hopefully suggested the benefits of the analogy noted throughout this report between the problem of diagnosis and that of reconfiguration. We next explore strategies which exploit this parallel to provide an integrated approach to FDIR.

Chapter 6

Integrating Diagnosis and Reconfiguration

As previously noted, viewing the problem of reconfiguration as a close analogue of the problem of diagnosis suggests the possiblity of exploiting diagnosis algorithms for reconfiguration, and of providing a unified basis for integrating the components of FDIR. In this chapter we explore the context for this integration and sketch algorithms to achieve it. We look first at a formal correspondence between diagnosis and reconfiguration, then at appropriate algorithms.

6.1 Mapping from Diagnosis to Reconfiguration

We have suggested that an efficient strategy for computing all reconfigurations is precisely that for computing all diagnoses, implying that a general diagnosis engine can be used for reconfiguration. To support this conjecture we specify an abstract engine and show how the specification can be interpreted to provide either a diagnosis or a reconfiguration engine, predicting that the transformation necessary to map a black box from a diagnosis engine to a reconfiguration engine is straightforward.

Let M be a domain model, including "normality" assumptions expressed in terms of a distinguished predicate P on components C, and let B be a specified (observed or desired) behavior. M and B are sets of first order formulas, and an explanation E is a set of subsets of C. Intuitively, E generalizes the notions of diagnosis and reconfiguration; the members of E "explain" the discrepancy, if any, between the model M and behavior B.

We define E to comprise satisfactory explanations relative to M and B just in case for all $\Phi \in E$,

$$M \cup B \cup \{P(c)|c \in \Phi\} \cup \{\neg P(c)|c \in C - \Phi\}$$

is consistent.[1]

The transformation necessary to map a diagnosis engine to a reconfiguration engine can be viewed as the interpretations that instantiate the general formulation, as given in Figure 6.1. Here **sd** and **sd′** are the system descriptions for diagnosis and reconfiguration, respectively, **obs** is the observed behavior, and **reqs** is the required or acceptable behavior. Δ_i is a diagnosis, and $\Re_{\Delta_i}$ is a reconfiguration.

Interpretation	M	B	P	E
Diagnosis	**sd**	**obs**	**ab**	$\{\Delta_1, \ldots, \Delta_n\}$
Reconfiguration	**sd′**	**reqs**	**rcfg**	$\{\Re_{\Delta_1}, \ldots, \Re_{\Delta_n}\}$

Figure 6.1: Interpretations of the mapping specification

Under the interpretation for diagnosis, the predicate P is the familiar abnormality predicate **ab**, which is used with negative polarity to express the normality assumptions, e.g., $\neg\mathbf{ab}(m_1)$ denotes that the component m_1 is behaving normally. Similarly, under the interpretation for reconfiguration, the normal assumption is that a component is not **reconfig**ured, denoted by $\neg$**rcfg**.

Note that if we consider a particular diagnosis engine, we can further specify the transformation as the composition or relative product of the operations for conflict generation and candidate recognition. In any case, given the simplicity of this mapping, it should in theory be possible to use an existing diagnosis engine for reconfiguration. We have of course said nothing about the degree of difficulty of such an enterprise; it seems reasonable to suspect that the more specialized the diagnosis engine, the harder the task. On the other hand, our analysis suggests the possibility of a general tool accommodating both diagnosis and reconfiguration in a single unified engine. In the next section we discuss algorithms for this approach.

[1] The characterization of satisfactory explanations by logical consistency is the crucial notion in the approach to diagnosis exploited here; for this reason, model-based diagnosis is often (and arguably more appropriately) referred to as "consistency-based" diagnosis.

6.2 Algorithms for Integrated FDIR

An ideal algorithm for integrated FDIR would simultaneously compute diagnoses and reconfigurations. There appear to be several reasons why this ideal is currently unobtainable. First, it would require a different characterization of reconfiguration than the one given above; if reconfigurations are defined relative to given diagnoses as we have specified, resulting strategies for integrated FDIR are inherently sequential because reconfiguration assumes an extant diagnosis. More generally, there is the question of how to handle the potentially conflicting nature of observations (**obs**) and requirements (**reqs**) in a consistency-based approach.

Nevertheless there are effective alternatives. One obvious approach is massive iteration: for each diagnosis and for each requirement, generate all possible reconfigurations. As an example of this approach, we modify Reiter's algorithm [40, p. 77] as follows. Instead of generating a pruned HS-tree as described, then returning the set of all minimal hitting sets (i.e., the set of all diagnoses), we perform the following steps *each time* we generate a minimal hitting set (i.e., a diagnosis Δ) and store a list of reconfigurations with respect to requirements for each diagnosis generated. Let $RQ = \{\mathbf{reqs}_1, \ldots, \mathbf{reqs}_n\}$ be the set of all reconfiguration requirements for the given system, and R be an initially empty set used to accumulate the current set of reconfigurations.

1. Let $RQ' := RQ$, $R := \emptyset$.
2. If there are no remaining requirements ($RQ' = \emptyset$) then return R to the diagnosis algorithm; else choose an element $REQS$ of RQ' and set $RQ' := RQ' - REQS$.
3. Generate the set R' of reconfigurations for the given Δ and $REQS$. Set $R := R \cup R'$. Go to 2.

There are several variations on this algorithm. A conservative approach would be to allow the integrated diagnosis/reconfiguration algorithm to terminate, then to sort the resulting list of reconfigurations with respect to desired criteria. For example, given the information returned, we could identify the class of diagnoses equivalent with respect to reconfiguration. Alternately, we could order the diagnoses with respect to cost of recoverability, for some measure of cost, and so on. Under certain circumstances a less conservative approach would be appropriate. For example, if time were critical, it might be acceptable to interrupt the diagnosis process immediately upon generation either of a nonempty set of reconfigurations, or of a set of one or more reconfigurations satisfying a minimal functionality requirement. These scenarios suggest a wide range of possible approaches to integrating diagnosis and reconfiguration. Clearly the less conservative approaches allow more substantive integration and thus more opportunity for reconfiguration factors to constrain diagnosis. It also seems reasonable to suspect that the benefits of general algorithms for integrating FDIR are ultimately limited; real optimization must also take into account the strategies of a particular diagnosis/reconfiguration engine. In any case, although the details of integrating diagnosis and recovery may vary, the benefits of integrated FDIR should now be clear. The rudimentary algorithms suggested in this chapter clearly represent directions for future work; effective algorithms which fully exploit the theoretical foundation presented in this report remain an interesting challenge.

Chapter 7

Limitations of the Model and Limits of the Analogy

In this chapter we explore the limitations of our work, focusing on the simplicity of our model and the limits of the suggested analogy between diagnosis and reconfiguration.

7.1 Limitations of the Model

The model of reconfiguration suggested in this report is very simple. A serious account of FDIR must factor in several dimensions including the level of redundancy, the level of acceptable functionality, and the granularity of the diagnosis versus that of the reconfiguration. Diagnosis associates abnormality with components, whereas reconfiguration potentially associates malfunction with a range of system units, the smallest of which is the diagnosable component. Representing the reconfiguration switches in the $\mathbf{sd}'$, as illustrated in some of the examples in Chapter 5, appears to be one way to accommodate the range of possibilities in a single general theory. Note that complete redundancy (i.e., standby sparing for all diagnosable components), granularity of reconfiguration identical to that of diagnosis, and requirements equivalent to original system functionality reduce FDIR to FD.

7.2 Limits of the Analogy: Minimality, Consistency, Entailment

Not surprisingly, the analogy we have pursued thus far has its limits. In this section we look at two particular points in the theory where the parallels between the problem of diagnosis and that of reconfiguration appear to weaken: the role of minimality in diagnosis and in reconfiguration, and the issue of consistency versus entailment.

As noted by [11], most earlier work in model-based diagnosis assumed a "superset property": any superset Δ' of a diagnosis Δ is also a diagnosis. The set of diagnoses can then be parsimoniously represented by the set of minimal diagnoses—those with no proper subsets that are also diagnoses. The algorithms of [40] and most early systems for consistency-based diagnosis construct only the minimal diagnoses and therefore rely on the superset property to ensure that they capture all diagnoses. The superset property can fail, however, with approaches that incorporate models of faulty, as well as correct, behavior.

Two approaches have been suggested for overcoming the inadequacy of minimal diagnoses in these cases [11]: one replaces the notion of minimal diagnosis with that of "kernel" diagnosis, the other places restrictions on the axioms that may appear in the system description so that the notion of minimal diagnosis remains adequate.

Our formulation of reconfiguration is similar to diagnosis with fault models in that the system description contains axioms describing behavior when a component is reconfigured, as well as when it is not. Thus, it is not surprising that reconfigurations do not have the superset property: for example, it is not acceptable to reconfigure (i.e., put the spare tire on) *both* the front and back wheels in our bike example.

The question then is: does loss of the superset property matter? Pragmatically, we do not think it does, for we surely prefer to reconfigure as few components as possible and will be satisfied if we can generate the minimal reconfigurations, without worrying about their supersets. Theoretically, though, the problem is more serious because the correctness arguments for Reiter's algorithm [40, pp. 67-68,77] and for the similar algorithm for reconfiguration depend on the superset property. While we do not yet have a definitive resolution for this difficulty, the following seems plausible.

The reason that the bike example fails to have the superset property is because we have the axiom

$$\neg rcfg(front) \vee \neg rcfg(back) \tag{7.1}$$

that explicitly rules it out. If we remove this axiom, we have a system description that satisfies a condition called LKAB [11] that is sufficient to ensure the superset property. We can therefore safely use Reiter's algorithm to generate all minimal reconfigurations relative to this revised system description. When we come to evaluate the candidate reconfigurations, we first filter them by condition (7.1).

We suspect that this technique may be quite widely applicable. For the (admittedly very few) examples we have considered so far, the system description can be encoded in axioms satisfying the LKAB condition, plus a few additional axioms that describe inadmissible combinations of reconfigurations that can be used as filters.

We believe that the issue of consistency versus entailment can be resolved by a postpass filter in a similar way. The point here is that by our definition, a satisfactory reconfiguration (relative to a diagnosis) is one that is consistent with the given model, the diagnosis, and the requirements. But is this an adequate characterization? Surely we want to know that the proposed reconfiguration is not merely consistent with the requirements, but will actually achieve (i.e., entail) them. We are sympathetic to this point of view but do not have a good way to satisfy it directly. However, assuming our logical system is sound, we can verify entailment by proving the theorem

$$\begin{aligned} &\mathbf{sd}' \cup \{\mathbf{ab}(c) | c \in \Delta\} \cup \{\neg\mathbf{ab}(c) | c \in \mathbf{comps} - \Delta\} \\ &\quad \cup \{\mathbf{rcfg}(c) | c \in \Re_\Delta\} \cup \{\neg\mathbf{rcfg}(c) | c \in \mathbf{comps}' - \Re_\Delta\} \\ &\vdash \ \mathbf{reqs} \end{aligned} \tag{7.2}$$

Thus, (7.2) can be added to superset constraints such as (7.1) as a further filter on acceptable reconfigurations. Note that if (7.2) is not a theorem, then the **sd** is surely rather weak, since it fails to adequately constrain the behavior of the system. A topic for further investigation is to determine whether constraints on the forms of axioms comprising the **sd** can be found that are sufficient to ensure entailment of requirements.

Chapter 8

Conclusions and Future Work

We have proposed a characterization of reconfiguration as an extension of Reiter's theory of model-based diagnosis. Our contribution has been to recognize and exploit the analogy between the problem of model-based diagnosis and that of model-based reconfiguration, yielding a unified basis for FDIR and a well defined context for integrated FDIR algorithms. We now look more closely at the implications of these developments.

8.1 Concluding Remarks

In the course of this report we have argued that previous work on automated fault diagnosis has focused almost exclusively on diagnosis, and that an integrated approach to FDIR is an essential next step if automated diagnosis is to address the requirements of practical applications such as air and space missions. What is the expected payoff of foundational work in reconfiguration and integrated FDIR such as that described here? We feel that once the focus is expanded from diagnosis[1] to FDIR, we can begin to realize and then to optimize the results of a comprehensive approach, in which the various facets of the task—diagnosis, identification, reconfiguration—mutually reinforce and constrain the outcome. An example of the type of benefit that can follow from an integrated approach to FDIR is a reduction in the fault

[1] Some of the work on diagnosis arguably includes fault identification as well. We are less interested in the precise delineation of the components of FDIR, than in a comprehensive approach to the problem.

discrimination problem. Typically there will be several candidate diagnoses for a given set of symptoms; techniques for discriminating among them form a major theme for much of the work in model-based diagnosis. However, if all candidates require the same recovery action, then there is no need to discriminate among them. In general, it is only necessary to discriminate among diagnoses to the extent that they require different recovery actions. Typically, the available recovery actions are rather limited, certainly fewer than the number of possible malfunctions, so discriminating on the basis of recovery action will generally be a simpler task than discriminating on the basis of additional testing or reevaluating criteria available at diagnosis time.

While a comprehensive approach would not have been viable a decade or so ago before the field of model-based diagnosis had matured, we feel that it is now definitely time to take an integrated view of FDIR, especially in applications where operating constraints imposed by physical systems in use mandate a balanced response to malfunction, i.e., the response most likely to allow the system to continue to function in a specifiably safe mode. The characteristics of air- and spacecraft provide a perspective on the diagnosis problem that differs from those employed in the traditional AI literature. Aside from the fact that we are dealing with machinery in operation, i.e., with operative diagnosis [1,2], we are also dealing with systems that typically provide considerable redundancy. The major characteristics of operative diagnosis is that faults must be diagnosed while the system is in operation, not on the test bench. Thus, failures may be masked by compensating control inputs, faults may be propagating while diagnosis is being performed, and it may be necessary to get the system into a safe state without necessarily having a solid diagnosis of the cause of the problem. The presence of redundancy, and the absence of opportunities for direct intervention, change the nature of the fault discrimination process. In the traditional domains of fault diagnosis, i.e., human physiology and electric circuits, discrimination among diagnoses is performed through serial tests and measurements. Much work has been done on techniques for minimizing the number of tests and on finding optimum probe points for measurements in circuits. On board a spacecraft, however, there is no opportunity to obtain new measurements; we are limited to the data provided by the in-place sensors. Instead, discrimination is performed by changing operating parameters (e.g., does the problem go away at reduced levels of thrust?) and by exploiting redundancy (e.g., does the problem go away when we switch to a backup unit?). Since redundancy also provides the main means for fault recovery, the problem of fault discrimination is intimately related to those of recovery and reconfiguration. It follows that both problems are probably best handled by uniform and integrated techniques.

A further reason for preferring an integrated approach to FDIR over separate techniques for each of its component problems concerns the trustworthiness of the resulting system. The proper measure of the trustworthiness of FDIR is the extent to which it recovers from faults and enables the mission to proceed in safety. The effectiveness of solutions to the subsidiary problems of diagnosis, isolation/identification and reconfiguration are significant only in terms of their contribution to the overall goal. A highly capable diagnostic system is of little use if it is harnessed to a poor recovery algorithm; the importance of occasional misdiagnoses can be evaluated only in terms of their consequences on overall system behavior. We believe that by concentrating on the total FDIR problem it will become possible to develop trustworthy systems for FDIR that can take proper account of the potential consequences of misdiagnosed faults and incorrect recovery actions.

We do not want to overstate our case; we are not claiming that integrated FDIR is the only option. It is certainly possible to view FDIR as a sequence of independent processes, possibly with enough communication to allow the constraints of one or more processes to influence the outcome of another. Yet it is hard to see how to achieve the effectiveness of an integrated approach within the sequential paradigm without significant duplication of effort or information. A further disadvantage of the sequential approach to FDIR is that it tends to impose a static definition of the problem. Conversely, one of the advantages of integrated FDIR is that it allows the distinction between fault detection, identification, and reconfiguration to be blurred, and, in the extreme, eliminated altogether. Thus, an integrated approach to FDIR also provides a more fertile context for new ideas in FDIR.

8.2 Future Work

As the preceding comments suggest, there are several interesting directions for future research. We have provided a foundation for integrated FDIR and have begun exploring strategies for integrating diagnosis and reconfiguration. In the future we would like to carry out a more detailed study of algorithms for integrated FDIR, based on the approach presented here but extended to accommodate recent collaboration by de Kleer, Mackworth, and Reiter [10], that extends and in some cases corrects Reiter's characterization of diagnosis. In addition, as we refine the notion of integrated FDIR for real-time and operative applications, we need to consider introducing a notion of time into our algorithms. We would also like to explore the connections between our work and Friedrich's therapeutic approach [18]; our intuition is that Friedrich's notions can be interpreted as a special case of our approach, but this conjecture clearly requires substantial justification. Finally, we would like to use one or more existing diagnosis systems to continue experimental evaluation of the use of diagnostic engines for reconfiguration; our initial attempt to use an existing diagnosis system to mechanize the light bulb example discussed in Chapter 5 was successful [24], suggesting the utility of further experimental efforts.

Bibliography

[1] K. Abbott. Robust Operative Diagnosis as Problem Solving in a Hypothesis Space. In *Proceedings, AAAI88*, pages 369–374, St. Paul, MN, 1988.

[2] K. Abbott. *Robust Fault Diagnosis of Physical Systems in Operation.* PhD thesis, Rutgers, The State University of New Jersey, 1990.

[3] P. Bonissone and P. Halverson. Time-constrained reasoning under uncertainty. *Real-Time Systems*, 2(1,2):25–45, May 1990.

[4] J. S. Brown, R. Burton, and J. de Kleer. Pedagogical, natural language, and knowledge engineering issues in SOPHIE I, II, and III. In D. Sleeman and J. S. Brown, editors, *Intelligent Tutoring Systems*, pages 227–282. Academic Press, New York, NY, 1982.

[5] B. Chandrasekaran and S. Mittal. Deep Versus Compiled Knowledge Approaches to Diagnostic Problem-Solving. *International Journal on Man-Machine Studies*, 19:425–436, 1983.

[6] P.T. Cox and T. Pietrzykowski. General diagnosis by abductive inference. In *Proceedings, IEEE 1987 Symposium on Logic Programming*, pages 183–189, San Francisco, CA, August 1987.

[7] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1-3):347–410, December 1984.

[8] R. Davis and W. Hamscher. Model-based reasoning: Troubleshooting. In H. E. Shrobe, editor, *Exploring Artificial Intelligence*, chapter 18, pages 297–346. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988. (Survey Talks from the National Conferences on Artificial Intelligence).

[9] J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24(1-3):7–83, December 1984.

[10] J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses. In *Proceedings, AAAI 90 (Volume 1)*, pages 324–330, Boston, MA, July 1990.

[11] J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. Technical Report SSL-90-40, Xerox Palo Alto Research Center, 1990.

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

[13] J. de Kleer and B. C. Williams. Diagnosis with behavioral modes. In *Proceedings, 11th IJCAI*, pages 1324–1330, Detroit, MI, August 1989.

[14] K. Decker, V. Lesser, and R. Whitehair. Extending a blackboard architecture for approximate processing. *Real-Time Systems*, 2(1,2):47–79, May 1990.

[15] S. Edwards and A. Caglayan. Expert systems for real-time monitoring and fault diagnosis. Technical Report CR 179441, NASA, 1989. U.S. Department of Commerce NTIS No. N89-23209.

[16] K.D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24(1-3):85–168, December 1984.

[17] K.D. Forbus. Qualitative physics: Past, present, and future. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, chapter 1.1, pages 11–39. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[18] G. Friedrich, G. Gottlob, and W. Nejdl. Hypothesis classification, abductive diagnosis and therapy. In G. Gottlob and W. Nejdl, editors, *Expert Systems in Engineering*, pages 69–78. Springer-Verlag Lecture Notes in Artificial Intelligence Vol. 462 (International Workshop Proceedings), Vienna, Austria, September 1990.

[19] G. Friedrich, G. Gottlob, and W. Nejdl. Physical impossibility instead of fault models. In *Proceedings, AAAI 90 (Volume 1)*, pages 331–336, Boston, MA, July 1990.

[20] G. Friedrich, G. Gottlob, and W. Nejdl. Formalizing the repair process. In *Proceedings, Second International Workshop on Principles of Diagnosis*, Milano, October 1991. To Appear.

[21] M.R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3):411–436, December 1984.

[22] R. Greiner, B. Smith, and R. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41(1):79–88, November 1989.

[23] W. Hamscher. Temporally coarse representation of behavior for model-based troubleshooting of digital circuits. In *Proceedings, 11th IJCAI*, pages 887–893, Detroit, MI, August 1989.

[24] W. Hamscher. Private communication, March 1991.

[25] W. Hamscher and R. Patil. Model-based diagnosis. Tutorial MA1 of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89, Detroit, MI, August 1989.

[26] A. Howe, D. Hart, and P. Cohen. Addressing real-time constraints in the design of autonomous agents. *Real-Time Systems*, 2(1,2):81–97, May 1990.

[27] H. Kautz. A formal theory for plan recognition. Technical Report TR-215, Department of Computer Science, University of Rochester, 1987.

[28] R. Korf. Depth-limited search for real-time problem solving. *Real-Time Systems*, 2(1,2):7–24, May 1990.

[29] B. Kuipers. Qualitative simulation. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, chapter 2.6, pages 236–260. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[30] J. Lark, L. Erman, S. Forrest, K. Gostelow, and F. Hayes-Roth. Concepts, methods, and languages for building timely, intelligent systems. *Real-Time Systems*, 2(1,2):127–148, May 1990.

[31] W. Nejdl. Belief revision, diagnosis and repair. In *Proceedings of the International Conference on Knowledge-Based Systems*, Munich, October 1991. To Appear.

[32] H. T. Ng. Model-based, multiple fault diagnosis of time-varying, continuous physical devices. In *Proceedings, 6th IEEE Conference on AI Applications*, pages 9–15, Santa Barbara, CA, 1990.

[33] J. Pan. Qualitative reasoning with deep-level mechanism models for diagnoses of mechanism failures. In *Proceedings, 1st IEEE Conference on A.I. Applications (CAIA-84)*, pages 295–301, Denver, CO, 1984.

[34] R. Patil, P. Szolovits, and W. Schwartz. Causal understanding of patient illness in medical diagnosis. In *Proceedings, 7th IJCAI*, pages 893–899, San Mateo, CA, 1981. Morgan Kaufmann Publishers, Inc.

[35] D. Poole. Normality and faults in logic-based diagnosis. In *Proceedings, 11th IJCAI*, pages 1304–1310, Detroit, MI, August 1989.

[36] H. Pople. Heuristic methods for imposing structure on ill-structured problems: The structuring of medical diagnosis. In P. Szolovits, editor, *Artificial Intelligence in Medicine*, pages 119–190. Westview Press, Boulder, CO, 1982.

[37] G. Provan and D. Poole. The utility of consistency-based diagnostic techniques. In *Proceedings, KR91*, pages 461–472, Cambridge, MA, April 1991. 2nd International Conference on Principles of Knowledge Representation and Reasoning.

[38] J. Reggia, D. Nau, and P. Wang. Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies*, 19(5):437–460, November 1983.

[39] R. Rehbold. Model-based knowledge acquisition from structure descriptions in a technical diagnosis domain. Technical Report SR-89-01, University of Kaiserslautern Department of Computer Science, 1989. Also in Proceedings of the *Specialized Conference on Second Generation Expert Systems* at the *9th International Workshop on Expert Systems and their Applications*, Avignon 1989, p. 193-205.

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

[41] F. Schneider. Critical (of) issues in real-time systems: A position paper. Technical Report 88-914, Cornell University Department of Computer Science, May 1988.

[42] E. Shortliffe. *MYCIN: Computer-Based Consultations in Medical Therapeutics.* American Elsevier, New York, 1976.

[43] H. Simon. *The Sciences of the Artificial.* The M.I.T. Press, Cambridge, 1969.

[44] P. Struss and O. Dressler. "Physical negation"—integrating fault models into the general diagnostic engine. In *Proceedings, 11th IJCAI*, pages 1318–1323, Detroit, MI, August 1989.

[45] B. C. Williams. Temporal qualitative analysis: Explaining how physical systems work. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, chapter 2.3, pages 133–177. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.