Issues in Diagnosis, Supervision and Safety

L. Nielsen
Email: lars@isy.liu.se

M. Nyberg
Email: matny@isy.liu.se

E. Frisk
Email: frisk@isy.liu.se

C. Bäckström
Email: cba@ida.liu.se

A. Henriksson
Email: andhe@ida.liu.se

I. Klein
Email: inger@isy.liu.se

F. Gustafsson
Email: fredrik@isy.liu.se

S. Gunnarsson
Email: svante@isy.liu.se

Abstract

Issues concerning diagnosis, supervision and saftey are found in many technologicaly advanced products. There is now a trend to extend the functionality of diagnosis and supervision systems to handle more advanced situations. This report collects some of the initiatives taking place in research and some of the developments taking place in the industry.

This work has been supported by NUTEK within the ISIS competence centre and by the Swedish research council for engineering sciences (TFR) under grant Dnr. 93-731.

IDA Technical Report 1996
LiTH-IDA-R-96-37
ISSN-0281-4250

Department of Computer
and Information Science,
Linköping University,
S-581 83 Linköping, Sweden

Contents

Chapter 1

Introduction

Diagnosis, supervision and safety are found in almost all technologically advanced products. This includes automobiles, airplanes, robots, numerically controlled machines, among others. There is now a trend to extend the functionality of diagnosis and supervision systems to handle more cases in more operating situations. There are many reasons including economy, safety, and maintenance.

The purpose of this report is to collect some of initiatives taking place in research and some of the developments taking place in industry.

1.1 Problem formulation

In [1989] Isermann defines the diagnostic task as the determination of kind, location, size and time of a detected fault.

A term closely related to diagnosis is *FDI (Fault Detection and Isolation)* as used by Frank [1991], Patton [1994] and Chow & Willsky [1984] where

- **Fault detection**
  Detect when a fault has occured.

- **Fault isolation**
  Isolate the fault, i.e. determine the faults origin

FDI is sometimes used as a synonym to diagnosis, e.g. in Gertler [1991].

When designing a diagnostic system important parameters are the false alarm rate, i.e. how often the system signals a fault in a fault-free environment, and the probability for missed fault detection. These measures can be hard to determine forcing other performance measures as will be discussed in Section 3.10.

To perform diagnosis we need some sort of *redundancy* in the system and one way of achieving this is to introduce *hardware redundancy* in the process. A critical component, e.g. an actuator or sensor, is then duplicated or triplicated (Triple Modular Redundancy) and then using a majority decision rule any faults in the

duplicated hardware can be detected. Hardware redundancy is straightforward to implement but has several drawbacks.

- Extra hardware can be very expensive.
- The extra hardware can be space consuming which can be of great importance, e.g. in a space shuttle. The components weight can also be of importance.
- Some components can't be duplicated, e.g. in a system to detect leaks on a pipeline it is not possible to duplicate the pipeline.

Instead of hardware redundancy we can utilize the system property *analytical redundancy* which are the subject of this chapter and can be defined as

Definition 1.1 ***[Analytical redundancy].*** *A process is analytically redundant if there exists functional relationships between measured or known variables, e.g. control signals.*

In [Chow and Willsky, 1984] analytical redundancy is said to exist in two forms

- *Direct* or *Static redundancy*
  The relationship among instantaneous outputs of sensors.
- *Temporal redundancy*
  The relationship among histories of sensor outputs and actuator inputs. It is based on these relationships that outputs of (dissimilar) sensors (at different times) can be compared.

When the system model is given as analytical functions, analytical redundancy is sometimes referred to as *functional redundancy*. One area where analytical redundancy based diagnosis will have problems replacing hardware redundancy is where the demands on fast reliable responses is very high, e.g. in an aircraft where human life could depend on extremely fast response to component failure.

The faults acting upon a system can be divided into three types of faults.

1. *Sensor (Instrument)* faults
   Faults acting on the sensors
2. *Actuator* faults
   Faults acting on the actuators
3. *Component (System)* faults
   A fault acting upon the system or the process we wish to diagnose.

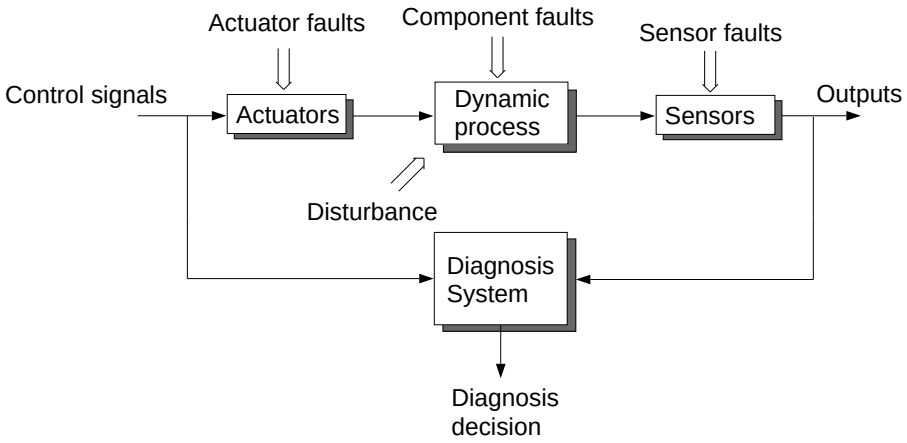

Figure 1.1: Structure of a diagnosis system

A general FDI scheme based on analytical redundancy can be illustrated as in Figure 1.1, an algorithm with measurements and control signals as inputs and a fault decision as output.

It may be unrealistic to assume that all signals acting upon the process can be measured, therefore an important property of an algorithm is how it reacts upon these *unknown inputs*. An algorithm that continue to work satisfactory even when unknown inputs vary is called *robust*. It is desirable to make the fault decision insensitive or even invariant to these unknown inputs, i.e. to perform exact or approximative disturbance decoupling. Further discussions around robustness issues can be found in section 3.2.

There are many ways to categorize the different diagnosis schemes described in literature, but here we divide them into two groups: *qualitative* approaches, emerging from the computer science field of studies, and approaches based on signal processing, control theory etc. here called *quantitative* approaches.

1.2 Outline of the report

Chapter 2 describes some industrial perspectives on diagnosis. It was inspired by the joint industrial and academic ISIS symposium on diagnosis, supervision and safety in March 1996, but has also been extended during further discussions with industrial partners. The chapters following reviews quite a number of possible techniques from the research literature. In Chapter 3 a continuous model based approach is described, and Chapter 4 deals with the problem of change detection. Chapter 5 is devoted to discrete model based diagnoses, while temporal reasoning is discussed in Chapter 6.

Bibliography

[Chow and Willsky, 1984] E.Y. Chow and A.S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Trans. on Automatic Control*, 29(7):603–614, 1984.

[Frank, 1991] P.M. Frank. Enhancement of robustness in observer-based fault detection. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 99–111, Baden-Baden, Germany, 1991.

[Gertler, 1991] J. Gertler. Analytical redundancy methods in fault detection and isolation-survey and synthesis. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 9–21, Baden-Baden, Germany, 1991.

[Isermann, 1989] R. Isermann. *Process fault diagnosis based on dynamic models and parameter estimation methods*, chapter 7. In Patton et al. [1989], 1989.

[Patton *et al.*, 1989] R.J. Patton, P. Frank, and R. Clark, editors. *Fault diagnosis in Dynamic systems*. Systems and Control Engineering. Prentice Hall, 1989.

[Patton, 1994] R.J. Patton. Robust model-based fault diagnosis:the state of the art. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 1–24, Espoo, Finland, 1994.

Chapter 2

Industrial perspectives

There are more issues involved in industrial diagnosis, supervision and safety than can be covered in this text. A sample of industrial perspectives are given in the following sections. These samples are based on contacts within ISIS (Information Systems for Industrial control and Supervision) both from a joint industrial and academic symposium in March 96 and from further contacts with the industries involved in ISIS.

2.1 Diagnosis, Supervision and Safety in process industry from an ABB perspective

ABB Industrial Systems AB develop, manufacture and sell control system products to the process industry, for example, the pulp and paper industry, chemical industry, breweries, food industry and metal industry. One part of ABB Industrial Systems is also dealing with motors, both AC and DC. The type of control system products manufactured by ABB Industrial systems include operator stations, controllers, batch stations, information management stations and engineering stations.

The situation today

Currently diagnoses is used on a component basis, i.e., each motor, pump, valve, transformer and so on is treated individually. The diagnoses consists of either localization of the fault after failure, or a detection algorithm giving an alarm for a possible fault. There are many problems with the approach used today.

The localization of a fault after failure is done completely off-line, and is totally separated from the control system. Often expensive sensors are used and complicated signal processing is necessary. This leads to the fact that the results can be understood only by a specialist.

One problem with the detection algorithms used today is that too many alarms are created, and some of these alarms are false alarms. Furthermore, it is difficult and very expensive to design these detection algorithms, since it is done individually for each process. The customers are not willing to pay this much for a fault detection algorithm. Additionally, considerable knowledge about a process is gained during the first to years when running the process, and during this time the industry learns what will cause them problems. This implies a great need for changes in the algorithms when the system is up and running. Often the industry is not willing to take these risks and costs, which in turn hampers good alarm systems.

The future

The operators want information on

1. what to do to reduce the impact on production when a failure has occured,
2. what to do to remove the failure, and
3. how to put back production to normal again.

To be a control system vendor supplying tools for this is a big challenge. We must make the functions easy to configure, validate, not CPU-demanding and always giving the correct information.

In the future we expect a development towards integrated diagnoses systems that use information from more than one component in order to make conclusions on faults using redundant information. ABB Industrial Systems has investigated methods like *Diagnostic Model Processor*. There are several benefits with this method. It is possible to point out what is failing with a high degree of certainty, and avoid false alarms. It also gives a possibility to suggest actions to eliminate the failure. The drawback is that this method is very costly in configuration and validation.

2.2 Diagnosis, Supervision and Safety in automotive engines

Diagnosis of automotive engines has become increasingly important, mostly because of legislative regulations. Today it is one of the major application areas for diagnosis, and the number of diagnosis systems in use is larger than for any other application involving mechanics. Compared to many other applications automotive diagnosis is constrained by economical reasons. Even the slightest costs gets emphasized because of the large production volumes.

Background

Diagnosis of automotive engines has a long history. Since the first automotive engines in the 18:th century, there has been a need for finding faults on the engines. For a long time, the diagnosis was performed manually, but diagnostic tools started to appear in the middle of the 20:th century. One example is the stroboscope that is used for determining the ignition time. In the 1960s, exhaust measurement became a common way of diagnosing the fuel system. Until the 1980s, all diagnosis were performed manually and off-board. It was around that time, electronics and gradually microprocessors were introduced in cars. This opened up the possibility to use on-board diagnosis. The objective was to make it easier for the mechanics to find faults. 1988, the first legislative regulations regarding On-Board Diagnostics, OBD, were introduced by CARB (California Air Resource Board). In the beginning these regulations applied only to California, but EPA (Environmental Protection Agency) adopted similar regulations that applied for all USA. This enforced the manufacturers to include more and more on-board diagnosis capability in the cars. 1994, the new and more stringent regulations, OBDII, were introduced in California. Today, software for fulfilling OBDII is a major part of the engine management system. At least 50% has been reported. Except for California and USA, few regulations have been introduced in other countries. However, for example EU have announced regulations, starting to apply in a few years.

Why On-Board Diagnosis?

There are several reasons for incorporating on-board diagnosis:

- The mechanics can check the stored fault code and immediately replace the faulty component. This implies more efficient and faster repair work.
- If a fault occurs when driving, the diagnosis system can, after detecting the fault, change the operating mode of the engine to *limp home*. This means that the faulty component is excluded from the engine control and a suboptimal control strategy is used until the car can be repaired.
- The engine can be repaired due to the condition of the engine and not due to a repair schedule, thus saving repair costs.
- The diagnosis system can make the driver aware of faults that can damage the engine, so that the car can be taken to a repair shop in time. This is a way of increasing the *reliability*.
- A fault can often imply increased emission of harmful emission components, dangerous for the environment. As an example, 1990 EPA estimated that

60% of the total hydro-carbon emissions originated from the 20% of the vehicles with serious malfunctioning emission control systems. It is important that such faults are detected so that the car can be repaired as quickly as possible.

The first three items can be summarized as to increase the *availability* of the car. Of all these reasons, the main reason for legislative regulations is the environmental issues.

OBDII

OBDII is the most extensive on-board diagnosis requirements announced so far. It started to apply 1994, but its requirements are made harder for every year until year 2000. The main idea is that a instrument panel lamp called Malfunction Indicator Light (MIL) must be illuminated in the case of a fault that can make the emissions exceed the emission limits by more than 50%. The MIL should, when illuminated, display the phrase "Check Engine" or "Service Engine Soon". The OBDII also contains standards for the *scantools*, connectors, communication, and protocols that are used to exchange data between the diagnosis system and the mechanics. Further, it says that the software and data must be encoded to prevent unauthorized changes of the engine management system.

The requirements on the diagnosis system is formulated so that it must be able to detect a fault during a *drive cycle*. A drive cycle is defined as a drive case where all characteristics of a *FTP75 test cycle* is present. FTP75 is a standardized test cycle used in USA and some other countries. When a fault occurs, the MIL must be illuminated. If the fault is still present the next drive cycle, a *Diagnostic Trouble Code* (DTC) and *freeze frame data* is stored. Freeze frame data is all information available of the current state of the engine and the control system. After three consecutive fault free drive cycles, the MIL should be turned off. Also, the fault code and freeze frame is erased after 40 fault free drive cycles.

Generally, the components that must be diagnosed in OBDII, is all actuators and sensors connected to the engine management system. Sensors and actuators must be limit checked to be in range. Also the values must be consistent with each other. Additionally, actuators must be checked using active tests. These general specifications apply therefore to for example mass air flow sensor, manifold pressure sensor, engine speed sensor, and the throttle. In addition to these general specifications, OBDII contains specific requirements and technical solutions for many components of the engine. Examples are:

- **Misfire**
  One of the most important parts of OBDII are the requirements regarding misfire. This is because a misfire means that unburned gasoline reach the catalyst, which can be overheated and severely damaged. The diagnosis system must be able to detect a single misfire and also to determine the

specific cylinder, in which the misfire occurred. During misfire, the MIL must be blinking.

The technology used today is signal processing of the RPM-signal. Sometimes an accelerometer is used as a complement. Also, ion current based methods are promising.

- **Catalyst**
  Another central part of OBDII is catalyst monitoring. The catalyst is a critical component for emission regulation. If the efficiency of the catalyst falls below 60%, the diagnosis system must indicate a fault. The technology used today is to use two lambda (oxygen) sensors, one upstream and one downstream the catalyst. For a fully functioning catalyst, the variations, due to the limit cycle enforced by the control system, in the upstream lambda sensor should not be present in the downstream sensor.

- **Lambda Sensors**
  A change in the time constant or an offset of the lambda sensors must be detected. This is done by studying the frequency, comparing the two sensors, and applying steps and studying step responses.

- **Purge System**
  The purpose of the purge system is to take care of fuel vapor from the fuel tank. It contains a coal canister and some valves to direct the fuel vapor from the tank into the canister and from the canister into the intake manifold. The diagnosis system must be able to detect malfunctioning valves and also a leak in the fuel tank. The technology used here is heavily based on active tests.

Other components for which OBDII contain detailed specifications are for example EGR-systems, fuel-systems, and secondary air systems.

2.3 Diagnosis, Supervision and Safety examples in AXE exchanges

This section briefly discusses diagnosis in the Ericsson AXE telephone exchange.

A telephone exchange is normally not considered a safety-critical system, although it could be considered so in certain cases, *eg.* it may be very important that a call for an ambulance succeeds without delay. Furthermore, the customers are demanding increasingly higher reliability from the products. For instance, telephone companies in Australia and the USA typically require that a telephone exchange is non-operational for at most 5 mins. per year, specifying economic penalties for the manufacturer if this requirement is not met.

Modern telephone exchanges, like the Ericsson AXE system, are complex systems consisting of interacting hardware and software. An AXE telephone exchange basically consists of the blocks shown in Figure 2.1. The software in an

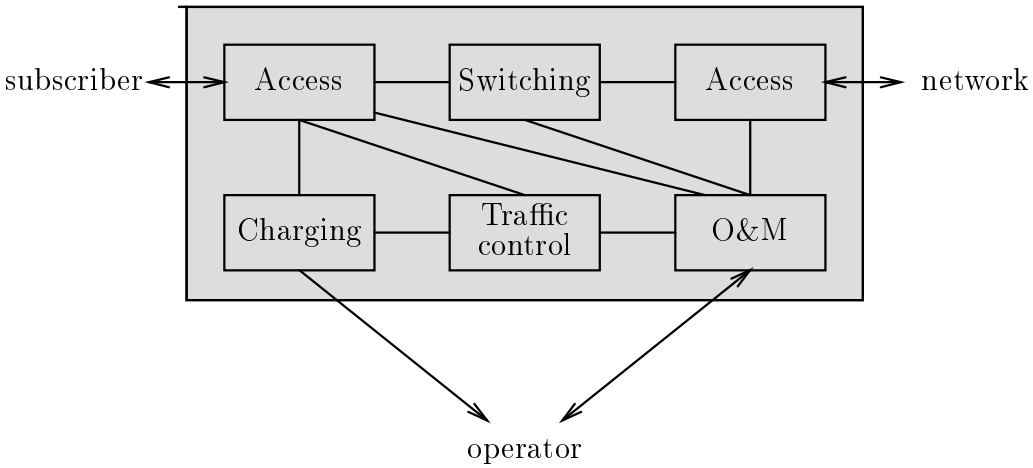

Figure 2.1: Block diagram of an AXE telephone exchange

AXE system contains several millions of lines of code (mostly written in Ericssons own application-specific language PLEX). Only some 10% of this code can be directly related to the main functionality of the system, *ie.* traffic management, charging and subscriber services. The remaining code is used for other purposes, including operating system, administration (*eg.* adding new subscribers), restart procedures, system extension, synchronization of systems *etc.* Most of these latter functionalities are located in the block labelled O&M (*operation and management*) in the figure. The O&M block consists of four subsystems:

MAS: Maintenance Subsystem (supervision of the hardware)

NMS: Network Management Subsystem (network load balancing)

STS: Statistical Subsystem (Collects statistics for number of calls, number of failed calls *etc.*)

RMS: Remote Measurement System

The functions of the NMS block can be divided into four different types, as follows:

Supervision: Raise an appropriate alarm when certain conditions are met (serious errors)

Observation: Change system state when required (for instance, in the case of system overload)

Control: There are two types of control actions:

- Protective control (*eg.* disallow certain types of calls in order to keep the system running)
- Expansive control (*eg.* find new paths for routing calls in the network)

Many different types of errors can arise in software-controlled telephone exchanges, for instance the following:

- Bit errors
- Sporadic hardware errors, *eg.* errors caused by static electricity (single or infrequent such errors need not always be reported)
- Synchronization slip, *eg.* a clicking sound caused by missed information due to synchronization problems between exchanges
- Protocol errors, caused by different communication protocols in exchanges, for instance, when modern digital and old analog exchanges are interconnected (single errors need often not be reported).

Sporadic errors should only be reported when frequent. This is solved by employing a so-called "leaking-bucket algorithm", which is based on maintaining a counter as follows: Whenever an error occurs, increment the counter by one. Decrease the counter by some fixed amount, larger than one, at certain predefined intervals. Raise an alarm whenever the value of the counter exceeds a preset limit. The parameters, *ie.* the value to subtract and the limit, are determined empirically after installation. However, the designers do not receive much feedback on how these parameters are set or how frequent alarms are in practice.

Many errors can also be attributed to the interfaces between system modules.

The AXE exchange collects a lot of statistics when operational. However, this statistics is seldom used and it seems not quite clear what statistics is relevant to use as feedback to the designers. A more intelligent way of collecting and interpreting statistics is desired.

2.4 Diagnosis, Supervision and Safety from a Saab Military Aircraft Point of View

Introduction

The work with flight safety and supervision are of very high priority at Saab Military Aircraft mainly since one single failure can cause the loss of an aircraft and human lives. There is also high priority in keeping the time the aircrafts are

grounded or not operational as short as possible by detecting and isolating faulty equipment in the aircraft.

By showing the general framework for setting the demands on every part of the aircrafts systems and giving two examples of how this can be achieved, we hope to give a view of how SMA works with these kinds of problems.

Risk of Aircraft Loss

The customer has specified a maximum number of aircraft losses per hour of flight and this number forms the basis for the work.

It is specified that 50% of these losses are allowed to be caused by technical problems and among these 50% one estimates that 50% can be caused by unknown technical problems leaving us with 25% of the maximum number of failures causing a loss of aircraft. This number is then divided in different parts forming a requirement for each system.

The risk of losing the aircraft is determined for every type of fault in each system and the probability of the fault is determined. These two numbers multiplied with each other and summed over every known fault for a system forms that systems contribution to the risk for an aircraft loss per flight hour.

Failure Mode Effect Analysis (FMEA) and Failure Tree Analysis (FTA) is used to predict the probabilities and the effects for all types of failures.

General Approach

To be able to keep the number of failures during flight down to an acceptable level a sophisticated supervision and diagnosis methodology is used where all systems have a Built-In-Test (BIT) using continuous monitoring during normal operation. It also includes Safety Check at each power on, self diagnosis, and test functions executed when a failure is detected or at predetermined intervals. Many parameters are also stored at a fairly high rate during each flight making it possible to do trend checking and to thoroughly investigate failure behavior.

Flight Control System

With the development of the fighter aircraft JAS39 Saab Military Aircraft took a further step towards high maneuverable aircraft but at the same time raised the risk of a crash in case of an undetected failure in the flight control system. The development of the flight control system has then been aimed at keeping the probability of an undetected failure down.

This has been achieved using an triplex redundant flight control system. The redundancy includes the sensors, the computers, and the actuators including redundancy in hydraulic and electrical power. A simple voting approach is used to determine which sensor and which command shall be used. Since there must

be physical redundancy in case of a failure this approach is very fast in detecting faults.

Since the supervision of the flight control system forms an important part of the safety of the aircraft a lot of emphasize is put into verifying the functionality of the system. This in addition to conventional software development testing also achieved by using simulators with real hardware but simulated sensors, actuators and flight dynamics. Different kinds of failures can then be introduced during simulated flight and the effects on the flight control can be evaluated.

Integrated Navigation System

There is a trend towards better and better position determination methods but to be able to use the achieved accuracy for other things than weapon delivery a fast and reliable fault detection and supervision methodology have to be implemented.

An aircraft navigation system typically consist of an inertial navigation system aided by GPS, Doppler radar, Terrain Referenced Navigation *etc.*. The systems are integrated using a kalman filter forming an analytic redundancy which can be used for model-based fault detection.

2.5 Diagnosis, Supervision and Safety examples in robotics

High productivity and availability are important issues for industrial robots. The productivity is determined by factors like the precision of the robot operation and the speed by which the robot is able to operate, while the availability depends of the overall operation of the robot and its components.

In order to improve the productivity there is big interest in developing the robot control system towards higher precision. One limitation for what can be achieved is determined by the quality of the mathematical model that is used for the design of the robot control system. It is therefore of interest to study methods that reduce the effects of modeling errors as much as possible. One approach to this problem is to use identification to, for example, determine parameters that are difficult to determine using physical modeling. A second approach is to utilize that robots in many situations carry out the same operation repeatedly, and add a correction to the control signal in order to improve the performance.

For high availability it is also important to have methods to detect, or even predict, and isolate different types of faults that can occur. Each minute that a production line has to be stopped represents a large economical loss. It is therefore of interest to develop method for efficient and reliable handling of error messages.

Chapter 3

Continuous model based diagnosis

3.1 Why model based diagnosis?

Why is there a need for a mathematical model to achieve diagnosis? It is easy to imagine a scheme where important entities of the dynamic process is measured and tested against predefined limits. The model based approach instead performs consistency checks of the process against a model of the process. There are several important advantages with the model based approach

1. Outputs are compared to their expected value on the basis of process state, therefore the thresholds can be set much tighter and the probability to identify faults in an early stage is increased dramatically.

2. A single fault in the process often propagate to several outputs and therefore causes more than one limit check to fire. This makes it hard to isolate faults without a mathematical model.

3. With a mathematical model of the process the FDI scheme can be made insensitive to unmeasured disturbances, e.g. in an SI-engine the load torque, making the FDI-scheme feasible in a much wider operating range.

4. It might be possible to perform the diagnostic task without installing extra sensors, i.e. the sensors available for e.g. control might suffice

There is of course a price to pay for these advantages in increased complexity in the diagnosis scheme and a need for a mathematical model.

3.2 Quantitative approaches to diagnosis

In quantitative approaches the diagnosis procedure is explicitly parted into two stages, the residual *generation* stage and the residual *evaluation* stage, as illus-

trated in Figure 3.1. The residual is a signal containing fault information, the

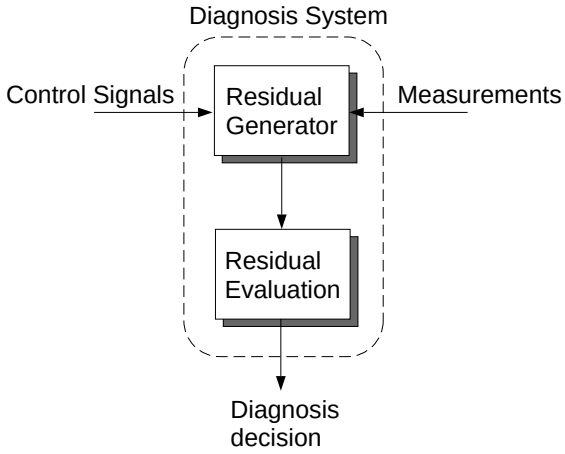

Figure 3.1: Two stage diagnosis system.

residual evaluation can in its simplest form be a thresholding test on the residual, i.e. a test if $|r(t)| > Threshold$. More generally the residual evaluation stage consists of a *change detection* test and a *logic inference system* to decide what caused the change. A change here represents a change in normal behavior of the residual.

The residual generation approaches can be divided into three subgroups, *limit & trend checking*, *signal analysis* and *process model based*.

- **Limit & trend checking**
  This approach is the simplest imaginable, testing sensor outputs against predefined limits and/or trends. This approach needs no mathematical model and are therefore simple to use but it is hard to achieve high performance diagnosis as was noted in section 3.1.

- **Signal analysis**
  These approaches analyses signals, i.e. sensor outputs, to achieve diagnosis. The analysis can be made in the frequency domain, [Neumann, 1991], or by using a *signal* model in the time domain. If fault influence are known to be greater than the input influence in well known frequency bands, a time-frequency distribution method as in [Olin and Rizzoni, 1991] can be used.

- **Process model based residual generation**
  These methods are based on a *process* model and will be further investigated in this chapter. The process model based approaches are further parted into two groups, *parameter estimation*, and *geometric* approaches. These methods will be investigated further, later in this chapter.

Before we can discuss the methods in this section we need to make some definitions. The approaches to be discussed here generates *residuals* which can be defined as

Definition 3.1 ***[Residual].*** *A residual (or parity vector) $r(t)$ is a scalar or vector that is 0 or small in the fault free case and $\neq 0$ when a fault occurs.*

The residual is a vector in the *parity space*. This definition implies that a residual $r(t)$ has to be *independent* of, or at least *insensitive* to, system states and unmeasured disturbances.

We will now concentrate on *linear* systems because they can be systematically analyzed, non-linear systems will be briefly discussed later in this chapter.

A general structure of a linear residual generator, can be described as in Figure 3.2. The transfer function from the fault $f(t)$ to the residual $r(t)$ then becomes

$$r(s) = H_y(s)G_f(s)f(s) = G_{rf}(s)f(s)$$

What conditions has to be fulfilled to be able to detect a fault in the residual?

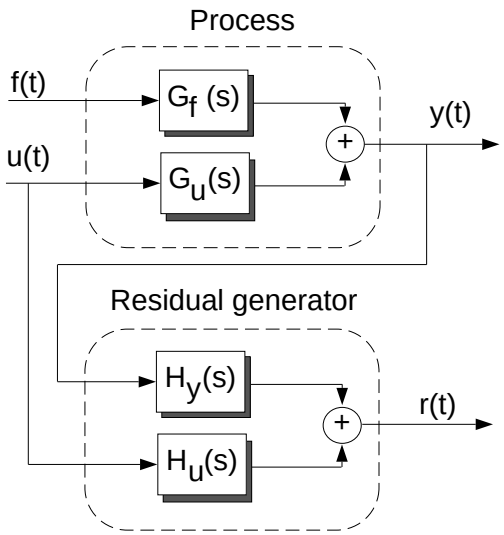

Figure 3.2: General structure of a linear residual generator

In [Chen and Patton, 1994] detectability has a natural definition. To be able to detect the i:th fault the i:th column of the response matrix $[G_{rf}(s)]_i$ has to be nonzero, i.e.

Definition 3.2 ***[Detectability].*** *The i:th fault is detectable in the residual if*

$$[G_{rf}(s)]_i \neq 0$$

This condition is however not enough in some practical situations. Assume that we have two residual generators with structure as in Figure 3.2. When

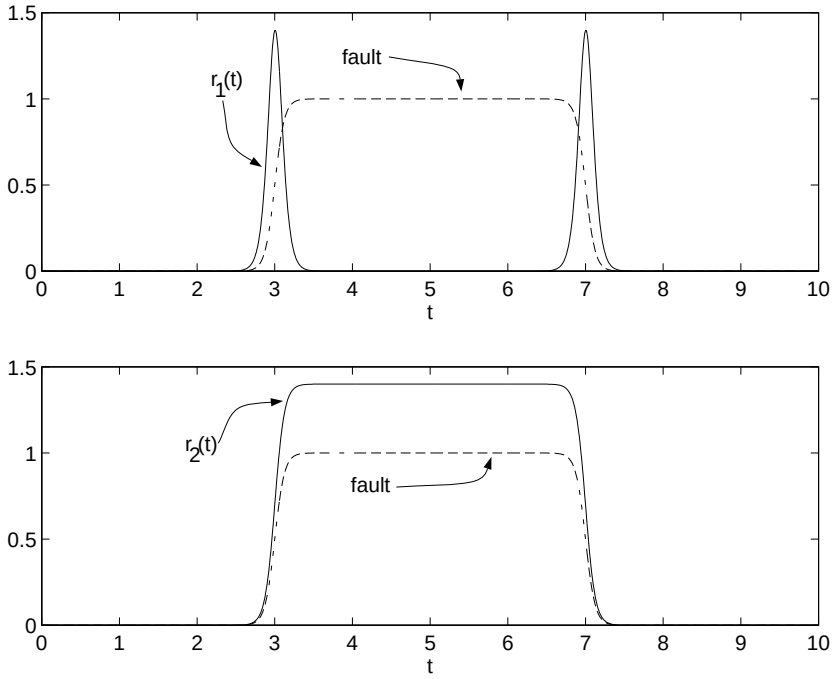

Figure 3.3: Example residuals

excited to a fault the residuals behave as in Figure 3.3. Here we see that we have a fundamentally different behavior between $r_1(t)$ and $r_2(t)$ as $r_1(t)$ only reflects changes on the fault signal and $r_2(t)$ has approximately the same shape as the fault signal. Thus $r_1(t)$ can not be used in a reliable FDI application even though it is clear that $G_{r_1 f}(s) \neq 0$.

The difference between the two residuals in the example are the value of $G_{rf}(0)$. It is clear that residual 1 has $G_{r_1 f}(0) = 0$ while residual 2 have $G_{r_2 f}(0) \neq 0$. This leads to another definition in [Chen and Patton, 1994]

Definition 3.3 ***[Strong detectability].*** *The i:th fault is said to be* strongly *detectable if and only if*

$$[G_{rf}(0)]_i \neq 0$$

The example show that it can be of great importance to perform a frequency analysis of the residual generator.

Note that in Definition 3, the frequency $\omega = 0$ is made particularly important. Which frequencies that is particularly important depends on which type of faults that are interesting. There are three different types of temporal fault behaviour as shown in Figure 3.4.

- Abrupt, step-faults a
- Incipient(developing) faults b

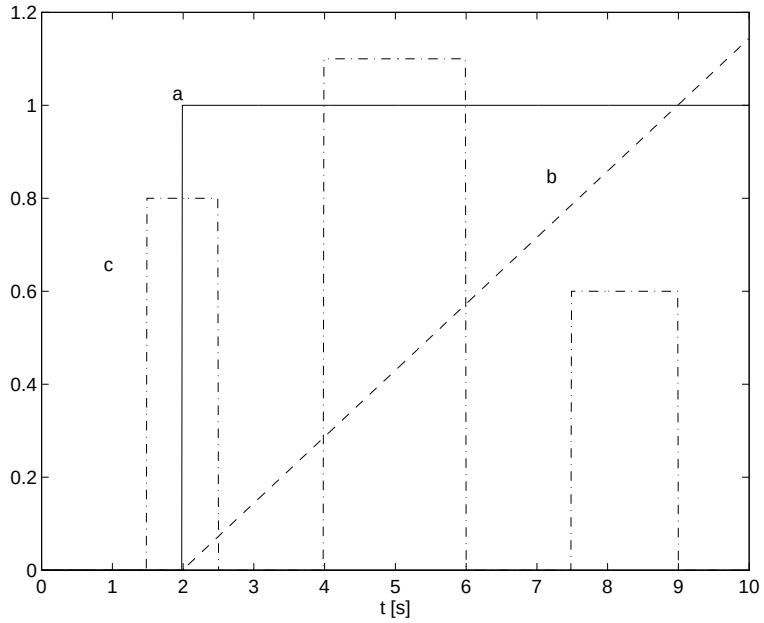

Figure 3.4: Different fault types

- Intermittant fault c

3.3 Isolation strategies

If we now have strongly detectable residuals, how can isolation be achieved? In [Patton, 1994] two general methods are described

- Structured residuals
- Fixed direction residuals

Structured residuals

The idea behind structured residuals is that a vector valued of residuals is designed making each element in the residual insensitive to different faults or subset of faults whilst remaining sensitive to the remaining faults, i.e. if we want to isolate three faults we can design a three dimensional residual with components $r_1(t)$, $r_2(t)$, and $r_3(t)$ to be *insensitive* to one fault each. Then if component $r_1(t)$ and $r_3(t)$ fire we can assume that fault 2 has occured.

Structured residuals can, e.g. be generated with a bank of observers. Here we will present the structure for *instrument fault diagnosis* (IFD), the corresponding structure for *actuator fault diagnosis* (AFD) and *component fault diagnosis*

(CFD) is trivial. There are two general structures for the observer bank, the *dedicated observer scheme* (DOS) or the *generalized observer scheme* (GOS). In DOS only one measurement is fed into each observer. The i:th observer are therefore only sensitive to sensor faults in the i:th sensor. DOS is illustrated in Figure 3.5. Each observer in a GOS scheme on the other hand are fed by all *but* one

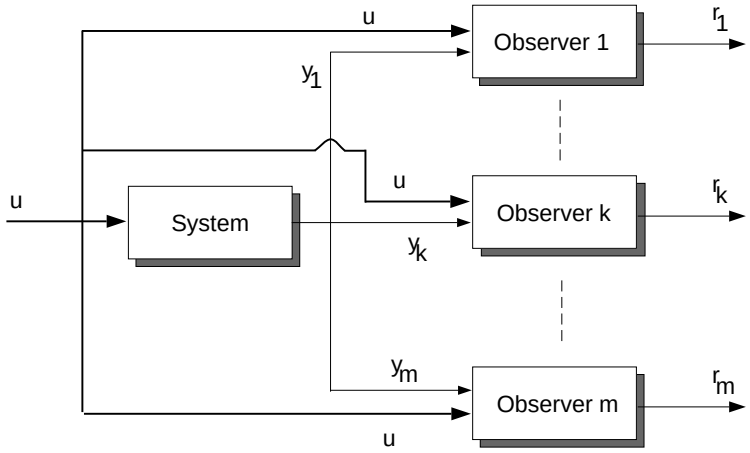

Figure 3.5: Dedicated Observer scheme for IFD

measurement making the i:th residual sensitive to all but the i:th measurement. GOS is illustrated in Figure 3.6. Since there always exists modelling errors and

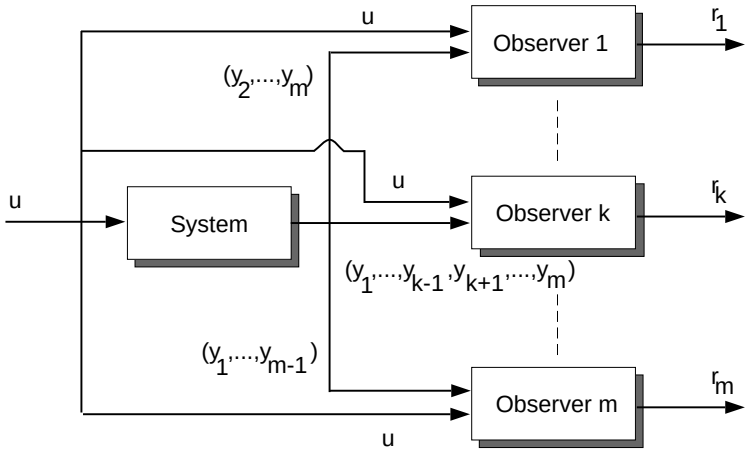

Figure 3.6: Generalized Observer scheme for IFD

disturbances not modeled, residuals are never 0 even in the fault free case. This can make some residuals fire that should not and vice versa. Therefore it is more likely that a GOS-bank of residuals are more reliable than a DOS-bank in a realistic environment. This is because that if one residual in a DOS-scheme happen to fire in a fault free case this immediately results in a bad fault decision. However in

I	f_1	f_2	f_3
r_1	1	1	0
r_2	1	1	1
r_3	1	1	1

II	f_1	f_2	f_3
r_1	1	1	0
r_2	1	0	1
r_3	1	1	1

III	f_1	f_2	f_3
r_1	0	1	1
r_2	1	0	1
r_3	1	1	0

Table 3.1: Example coding sets

a GOS-scheme more than half of the residuals have to misfire (if a majority decision rule is used) to make a bad fault decision. If a residual pattern, i.e. a binary vector describing which residuals that have fired, does not correspond to any fault patterns a natural approach is to assume the faultpattern that has the smallest Hamming distance to the residual pattern. The Hamming distance is defined as the number of positions two binary vectors differ, e.g. $\mathrm{d}((1,1,0),(0,1,1)) = 2$.

As always there is a price to pay for this increased reliability, or robustness, a GOS-scheme can only detect one fault at a time while a DOS-scheme can detect faults in all sensors at the same time. It is possible to extend a GOS scheme with extra sensors and residuals to achieve possibilities to detect and isolate multiple faults as in [Hsu *et al.*, 1995].

To illustrate how a bank of residuals are structured so called *coding sets* are used. In Table 3.1 three examples are presented and each row represents a residual, a 1 in position j on row i implies that fault f_j affects residual r_i. The different columns in the coding sets in the table is called the *fault code*. A coding set are a table that describes how different faults affect the residuals.

If for example in coding set III residuals r_1 and r_3 fire while r_2 does not, i.e. fault code $(101)^T$, it is probable that fault f_2 has occurred. To detect a fault, no column can contain only zeros and to achieve isolation all columns must be unique. If these two requirements are fulfilled, the coding set is called *weakly isolating*.

A small fault might fire some but not all elements in the residual vector that is sensitive to the specific fault. To prevent misisolation in these cases the coding set should be constructed so that no two columns can get identical when ones in a column are replaced by zeros. A coding set that fulfills this requirement is called a *strongly isolating* set.

In Figure 3.1 coding set I is non-isolating, II is weakly isolating and III is strongly isolating.

Fixed direction residuals

The idea with fixed direction residuals is the basis of the *fault detection filter* (FDF) where the residual vector get a specific direction depending on the fault that is acting upon the system.

Figure 3.7 gives an geometrical illustration of this type of residuals when a

fault of type 1 has occurred. The most probable fault can then be determined by

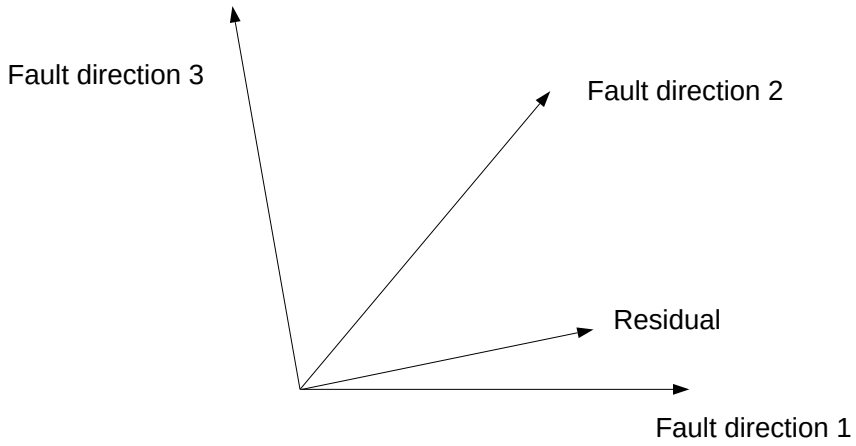

Figure 3.7: Fixed direction residuals

finding the fault vector that has the smallest angle to the residual vector.

It can be noted that a DOS scheme can be viewed as a fixed direction residual generator with the basis vectors as directions. A GOS scheme can however not be viewed as a fixed directions residual generator as a residual there is confined to a subspace of order $n-1$ (if the residuals has dimension n) instead of only a 1-dimensional subspace (the direction).

3.4 Robustness

As mentioned earlier, it is unrealistic to assume a perfect model and no disturbances acting upon the process. This makes the diagnostic task even harder, this problem is called the *robustness* problem and a diagnostic algorithm that continues to work satisfactory even when subjected to modeling errors and disturbances is called robust.

Since the ideal situation never occur in a real application, the robustness aspect is one of the most important issues when designing a diagnosis system. The methods to tackle the robustness problem can be divided into two categories [Frank and Ding, 1994]

- Robust residual generation, *active robustness*
- Robust residual evaluation, *passive robustness*

Robust residual generation

These methods strive to make the residuals insensitive or even invariant to model uncertainty and disturbances, and still retain the sensitivity towards faults. There

are two different types of disturbances, *structured* and *unstructured* disturbances. If it is "known" *exactly how* a disturbance signal influences the process it is called structured uncertainty and this high degree of disturbance knowledge is enough to actively reduce or even eliminate the disturbance influence on the residual. However if no knowledge of the disturbance is known, no active robustness can be achieved. Examples of robust generation methods are Unknown Input Observers (UIO)[Frank and Wünnenberg, 1989], Eigenstructure assignment of observers [Patton and Kangethe, 1989, Patton, 1994], robust parity relations [Chow and Willsky, 1984, Gertler, 1991].

Robust residual evaluation

The goal with robust evaluation methods is to enable reliable decision-making and still keeping the false-alarm rate satisfactorily small. Examples of robust evaluation methods are adaptive thresholds [Ding and Frank, 1991], decision making based on fuzzy logic [Frank, 1993], and statistical change detection methods (sometimes referred to as *statistical decoupling*).

3.5 Model structure

To proceed in the analysis of residual generation approaches we need an analytical model. In this report a state-representation of the model are used as

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t), u(t)) \end{aligned} \tag{3.1}$$

The linear (time-continuous) state representation is

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{3.2}$$

As we have noted earlier we have three general types of faults:

1. *Sensor (Instrument)* faults
   Modeled as an additive fault to the output signal.

2. *Actuator faults*
   Modeled as an additive fault to the input signal in the *system dynamics*

3. *Component (System) faults*
   Modeled as entering the *system dynamics* with any distribution matrix. Here it is seen that actuator faults only are a special case of component faults.

There are also uncertainties about the model or unmeasured inputs to the process, e.g. the load torque in an automotive engine. If these uncertainties are *structured*, i.e. it is known how they enter the system dynamics, this information can be incorporated into the model.

In the linear case and if model uncertainties are supposed structured, the complete model becomes

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B(u(t) + f_a(t)) + Hf_c(t) + Ed(t) \\ y(t) &= Cx(t) + Du(t) + f_s(t) \end{aligned} \tag{3.3}$$

where $f_a(t)$ denotes actuator faults, $f_c(t)$ component faults, $f_s(t)$ sensor faults and $d(t)$ disturbances acting upon the system. H and E is called the distribution matrices for $f_c(t)$ and $d(t)$.

3.6 Parameter estimation

As we noted in 3.2, process model based residual generators could be parted into two approaches parameter estimation and geometric approaches. A parameter estimation method, [Isermann, 1989, Isermann, 1991] is based on estimating important parameters in a process, e.g. frictional coefficients, volumes or masses, and compare them with nominal values.

We first need to define the model structure to use. The process to be modeled typically consist of both *static relations* and *dynamics relations*, both *linear* and *non-linear*.

Theoretically there is no limit on the appearance of these relations, the parameter estimation could be done by e.g. a straightforward gradient-search algorithm. But to enable efficient estimation of model parameters here it is assumed that the model is linear in its parameters. A least squares solution are then easy to extract. Note that this in no way implies a linear model. The equation

$$y(t) = a_1 \, x^2(t)$$

is linear in its parameter a_1 but is clearly non-linear.

With this assumption the model can be written as a linear regression model

$$y(t) = \varphi^T(t)\theta \tag{3.4}$$

where $\varphi(t)$ consists of inputs and old measured variables in a discrete model and output derivatives in a continuous model. θ are the model parameters to be estimated.

Note that θ is the *model* parameters, not the *physical* parameters. θ can be written as a function of the physical parameters p as

$$\theta = f(p) \tag{3.5}$$

Note that it can be of great importance how in- and out-signals are chosen as we will see in the example below.

Example 3.1. Consider a simple linear system, a first order low pass RC-link. Here there are two physical parameters, the resistance R and capacitance C.

If the input and output voltages, u_1 and u_2 are chosen as in and out signals, the system gets

$$u_2(t) = -RC\dot{u}_2(t) + u_1(t) = \varphi^T(t)\theta = (-\dot{u}_2(t)\ u_1(t)) \begin{pmatrix} RC \\ 1 \end{pmatrix} \tag{3.6}$$

In equation (3.6) we see that only one parameter appear in θ as RC. We can then conclude that the two parameters cannot be estimated with this choice of input-output signals. If we instead considers the output current i_2 as output signal the system gets:

$$i_2(t) = -RC\dot{i}_2(t) + C\dot{u}(t) = \varphi^T(t)\theta = (-\dot{i}_2(t)\ \dot{u}(t)) \begin{pmatrix} RC \\ C \end{pmatrix} \tag{3.7}$$

Here in (3.7) two parameters appear and both R and C are identifiable. In a practical problem there might not be a choice in in-out signals but the example shows that in a parameter estimation method, the in-out signal choice can be of great importance and should be analyzed.

Now when the model structure is defined we can outline the typical parameter estimation diagnosis method.

- **Data processng**
  With the help of the model and measured output data, model parameters can be estimated, e.g. by minimizing the quadratic estimation error

  $$V_N(\theta) = \sum_{i=0}^{N} \left(y(i) - \varphi^T(i)\theta\right)^2$$

  The LS-solution can easily be replaced by a RLS-estimator to achieve adaptability to a time varying process.

- **Fault detection**
  When an estimation of model parameters $\hat{\theta}$ is produced, an estimation of process parameters $\hat{p}$ can be extracted by inverting equation (3.5), this is also called *feature extraction.*

  $$\hat{p} = f^{-1}(\hat{\theta})$$

  Also $\Delta p = p_{nominal} - \hat{p}$ and standard deviation σ_p can be extracted to be used in a statistical test whether a fault is acting upon the system or not.

  Δp and σ_p can be seen as residuals as they are small in the fault-free case. They are also in parameter estimation articles called *syndromes.*

- **Fault classification**
  If the statistical test mentioned above decides that a fault is present, isolation of the fault source is the final stage in a parameter estimation method.

The algorithm outlined above is an example of a typical algorithm, another approach is taken in [Isermann, 1989] where the detection and classification steps are combined into one using a Bayes classification rule.

3.7 Geometric approach to residual generation

The approaches described in this section are called parity space approaches because they generate residuals who are vectors in the parity space. The methods can be divided into open- and closed-loop approaches. In an open-loop approach there are, as the name suggests, no feedback from previously calculated residuals.

The idea behind closed-loop approaches, i.e. observer based approaches, are to use a state-estimator as a residual generator. Both structured residuals and fixed direction isolation methods are achievable with both open- and closed-loop design methods. There are a number of approaches suggested in literature, here we will address

- State observers
- Fault detection filter
- Unknown Input Observers
  - By parity equations
  - By Kronecker canonical form
  - By eigenstructure assignment of observer

Note that these are *methods* to design the residual generator. Several of these designs may result in the same residual generator in the end as shown in [Gertler, 1991].

State observers

If there are no uncertainties acting upon the system, a straightforward approach is to use a state estimator observer and compare the estimated outputs with the measured.

Consider the special case of IFD. Assume a linear system with additive sensor faults f_s as

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du + f_s \end{aligned} \tag{3.8}$$

A state observer for system (3.8) can be stated as

$$\begin{aligned}\dot{\hat{x}} &= A\hat{x} + Bu + K(y - \hat{y}) \\ \hat{y} &= C\hat{x} + Du\end{aligned}$$

If $r = y - \hat{y}$ is used as the residual it can be written

$$r = y - \hat{y} = Cx + Du + f_s - C\hat{x} - Du = Ce + f_s$$

where e is the state estimation fault $e = x - \hat{x}$. The estimation error dynamics can be stated

$$\dot{e} = (A - KC)e - Kf_s$$

Assume f_s is a step from 0 to $F \neq 0$. Since $A_c = A - KC$ is a stable matrix, e will go towards a stationary value

$$e \to A_c^{-1}KF \text{ as } t \to \infty$$

As $r = Ce + f_s$ and e goes towards a non zero value the residual will be $\neq 0$ if $F \neq 0$ *and* $A_c^{-1}K + I \neq 0$. It can be seen that in a single-output system $A_c^{-1}K + I \neq 0$ is equivalent with $\det(A) \neq 0$.

Fault detection filter

The idea with the fault detection filter [Gertler, 1991, Patton, 1994] is, as was noted in earlier, to produce fixed direction residuals. The method is based on an observer of the form

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x} - Du)$$

Considering a fault in the $i : th$ actuator we get estimation error $e = x - \hat{x}$ dynamics as

$$\begin{aligned}\dot{e} &= (A - KC)e + b_i f_a \\ e_y &= y - \hat{y} = Cx + Du - C\hat{x} - Du = C(x - \hat{x}) = Ce\end{aligned}$$

where b_i is th $i : th$ column in B. By a special choice of K it is possible to make e_y, i.e. the residual, grow in a specified direction when the $i : th$ fault has occured.

An efficient design procedure including eigenstructure assignment of observer has been found but in [Patton, 1994]. It is noted that the fixed direction approach uses up more of the design freedom compared to other observer based approaches described next who therefore supersedes the fault detection filter.

Unknown Input Observers

If disturbances are acting upon the system or model uncertainties are prominant, robust methods has to be used. Robust observers is called *Unknown Input Observers*, and can be designed in a number of ways.

Parity equations from a state-space model

Parity equations is at first sight no observer based residual generator, but it can be shown [Patton and Chen, 1991] that discrete parity equations can be seen as a dead-beat observer. This approach will be described in detail later on in this chapter.

By Kronecker Canonical Form

By putting the system on a special form, an observer can be designed so that disturbance influence on the *state-estimate* can be eliminated
[Frank and Wünnenberg, 1989] and *robust* residuals can be generated. It is however not necessary to decouple disturbance influence in the state-estimate, only disturbance decoupling in the *output-estimate* is needed.

Eigenstructure assignment of observer

The eigenstructure assignment [Patton and Kangethe, 1989] is a method of designing identity observers, achieving disturbance decoupling in the residual.

3.8 Residual evaluation

Due to model uncertainties,measurement noise, and only approximate decoupling from unmeasured disturbances is achievable, residuals will not be 0 in the fault-free case. Therefore a non-zero threshold has to be selected. This is even more important in the case of unstructured uncertainties where exact disturbance decoupling in the residuals is impossible.

In [Frank, 1991] it is noted that when *deterministic* decoupling, i.e. decoupling of structured disturbances in the residuals, is not possible there is a possibility, if we know the statistical distribution of the residual, to use this knowledge and achieve robust FDI. This is called *statistical decoupling*.

One method who achieves statistical decoupling is the *GLR* (Generalized Likelihood Ratio)[A.S.—Willsky and Jones, 1974] test where the $k : th$ residual is modeled as

$$r_k(t) = r_{0,k}(t) + G_k(p)f(t)$$

where $r_{0,k}(t)$ is white noise with zero mean and G_k is the distribution matrix of the $k : th$ fault. p is the derivation operator, i.e. $\dot{y}(t) = p\,y(t)$.

A hypothesis test is then performed with the hypothesis

$$\begin{aligned} H_0 &: r_k = r_{0,k} \\ H_i &: r_k = r_{0,k} + G_{i,k}\, f_i \text{ the } i{:}th \text{ fault has occured} \end{aligned}$$

The hypothesis decision can be made through a test of the likelihood ratio

$$L_i = \frac{Pr(r_1, \ldots, r_n | H_i, f_k = \hat{f}_i)}{Pr(r_1, \ldots, r_n | H_0)}$$

Where $Pr(\cdot)$ denotes the density function of the underlying stochastic process. Since neither $\hat{f}_i$ nor the probability density function under assumption H_i is know these has to be estimated. This motivates the name Generalized Likelihood ratio.

The decision is then based on the rule

$$\begin{aligned} L_i &> T_i : H_i \text{ is assumed, i.e. the } i{:}th \text{ fault is assumed present} \\ L_i &< T_i : H_0 \text{ is assumed, i.e. no fault} \end{aligned}$$

The desired false alarm rate can be adjusted by choosing suitable thresholds T_i.

This approach can be easily illustrated on a one dimensional residual by Figure 3.8. Assume the observed value of the residual is r. Assume H_0 is the density

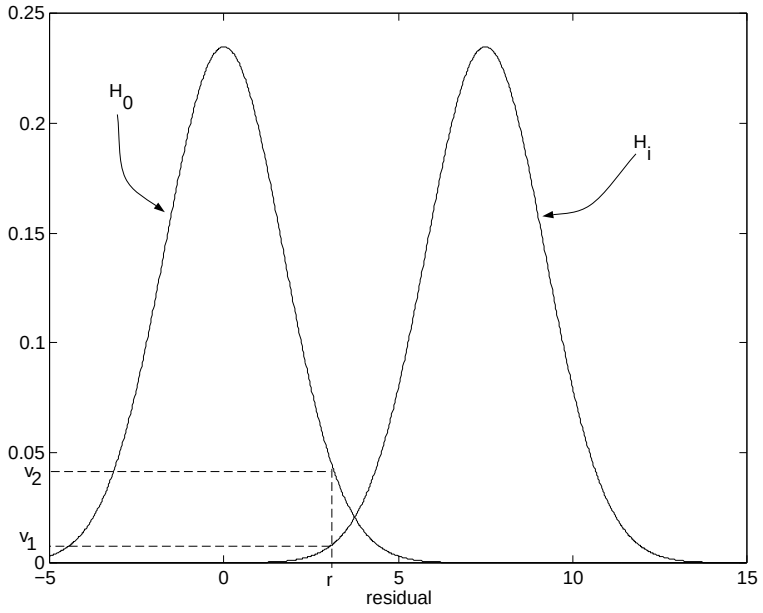

Figure 3.8: GLR illustration

function of r under assumption H_0 and H_1 is the density function of r under the assumption H_1. We can directly see that H_0 is the most probable hypothesis. L_i is then an *estimation* of $\frac{v_1}{v_2}$. In this example L_i would be small as $v_1 < v_2$ and hypothesis H_0 would be assumed, just as expected.

Another more intuitive approach to robust residual evaluation is that of *adaptive thresholds*. Since the model used does not model the system perfectly, the residuals will fluctuate with changing inputs even in a fault-free situation. There might be situations where these fluctuations are so great so that no threshold level fulfills both satisfactory false alarm rate demands and missed detection probabilities.

The adaptive thresholds approach is as noted above based on the fact that the residuals tend to fluctuate *with* the input signals (unmeasured or measured). Examples of adaptive thresholds can be that the threshold level is scaled with the size of the input vector, i.e. $T_i(t) \propto ||u(t)||$, or time-derivative of the input vector, i.e. $T_i(t) \propto ||\dot{u}(t)||$. Also fuzzy systems has been proposed [Frank, 1994] for residual evaluation.

In the end, we have to set the threshold levels. One simple approach is to observe the residuals in the fault free case and set the level to get the desired false-alarm rate. The residual evaluation rules used often get adapted to the application, e.g. by using time-limits on how long the residual can be above the threshold before a fault is assumed etc. It is easy to imagine a number of ad hoc solutions to improve robustness, but a systematic approach based on Markov theory choosing the thresholds has been suggested in [Walker, 1989].

3.9 Non-linear residual generators

As noted, all previously described residual generators are *linear*. When applying a linear residual generator, based on a linearization of a non-linear system, modelling errors can become dominant very quickly as the system deviates from the linearization point. One way to master this problem is to use a non-linear residual generator taking full advantage of the knowledge in the non-linear model. Non-linear residuals can be both closed-loop generators, [Frank, 1991], or open-loop generators [Krishnaswami and Rizzoni, 1994]. Non-linear parity equations is described in [Krishnaswami and Rizzoni, 1994].

In most applications it is not realistic to assume a linear model. In [Frank, 1993] a class of nonlinear systems are presented where decoupling is possible if the differential equations describing the system can be stated on the form

$$\begin{aligned} \dot{x} &= Ax + B(y,u) + E_1 d_1 + R_1 f \\ y &= Cx + E_2 d_2 + R_2 f + Du \end{aligned}$$

where d_1, d_2 are disturbance vectors and f are the fault vector. As the very special nonlinearity $B(y,u)$ only depends on measured variables, it can be compensated for by non-linear decoupling. This class of systems is very limited but e.g. industrial robots fits into this category.

As the above class is very limited, a larger class of non-linear systems where robust observer design has been successful is when the differential equations can

be written on the form

$$\begin{aligned} \dot{x} &= A(x) + B(x)u + E_1(x)d_1 + R_1(x)f \\ y &= C(x) + E_2 d_2 + R_2 f + Du \end{aligned}$$

3.10 Performance issues

What performance measures do we have to compare/evaluate different residual generators? Two natural measures are the *false alarm rate* and the *probability for missed detection*. It can however be difficult to design a diagnostic system based on these measures, especially the latter one who is hard to estimate. Instead a performance index can be defined that is used as an indicator of residual generator performance. Examples of performance indexes is given in [Gertler and Costin, 1994, Chow and Willsky, 1984, Patton, 1994]. The perfomance index π is often in the shape of

$$\pi = \frac{\text{fault-influence on the residual}}{\text{residual insecurity}}$$

where the denominator can e.g. be the variance of the residual in fault free operation and the numerator can be e.g. $|r(t)|$ when the residual is subjected to a fault. This performance index can be used for both optimization purposes and to compare different methods.

3.11 Parity equations

In this section parity equations [Gertler, 1991, Chow and Willsky, 1984] are described in detail and a design example is presented. Parity equations can be defined as consistency relations between inputs and outputs.

Consider the system:

$$y(t+1) = ay(t) + bu(t) + f(t)$$

In the fault-free case ($f(t) = 0$) the relation

$$y(t) - ay(t-1) - bu(t-1) = 0$$

should hold. By using the lefthandside of the relation we get a residual generator

$$r(t) = y(t) - ay(t-1) - bu(t-1)$$

It is easy to see that $r(t) = 0$ in the fault-free case and $r(t) \neq 0$ when $f(t) \neq 0$. This is an example of a parity equation. A systematic method of finding parity equations with desired properties is described below.

Method description

Restating the model given in equation (3.3), here a time-discrete form is used as it is more suited for this approach. First we consider the fault free, no disturbance case, i.e. $f_a = f_c = f_s = d \equiv 0$.

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{3.9}$$

It is not necessary to have the model on state-space form to develop the residual generator, it can just as well be developed using an input-output formulation of the model. The state-space form is chosen as it produces a clean notation.

Since we are going to utilize temporal redundancy we need an expression for the output based on previous states.
The output at time $t+1, t+2, \ldots, t+s, s > 0$ then becomes

$$\begin{aligned} y(t+1) &= CAx(t) + CBu(t) + Du(t+1) \\ y(t+2) &= CA^2x(t) + CABu(t) + CBu(t+1) + Du(t+2) \\ &\vdots \\ y(t+s) &= CA^sx(t) + CA^{s-1}Bu(t) + \ldots + CBu(t+s-1) + Du(t+s) \end{aligned}$$

Collecting $y(t-s), \ldots, y(t)$ in a vector yields

$$\mathbf{Y}(t) = \mathbf{R}x(t-s) + \mathbf{Q}\mathbf{U}(t) \tag{3.10}$$

where

$$\mathbf{Q} = \begin{pmatrix} D & 0 & \ldots & & 0 \\ CB & D & 0 & \ldots & 0 \\ CAB & CB & D & 0 & 0 \\ \vdots & \vdots & & \ddots & \\ CA^{s-1}B & CA^{s-2}B & \ldots & CB & D \end{pmatrix}$$

$$\mathbf{Y}(t) = \begin{pmatrix} y(t-s) \\ y(t-s+1) \\ y(t-s+2) \\ \vdots \\ y(t) \end{pmatrix} \quad \mathbf{U}(t) = \begin{pmatrix} u(t-s) \\ u(t-s+1) \\ u(t-s+2) \\ \vdots \\ u(t) \end{pmatrix} \quad \mathbf{R} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^s \end{pmatrix}$$

Assuming k inputs and m measurements vector $\mathbf{Y}$ is $[(s+1)m]$ long and $\mathbf{U}$ is $[(s+1)k]$ long. Matrix $\mathbf{R}$ has dimensions $[(s+1)m \times n]$ and $\mathbf{Q}$ has $[[(s+1)m] \times [s+1]k]$. Note that $y(t)$ and $u(t)$ are vectors and not scalar values.

In equation (3.10), $\mathbf{Y}$, $\mathbf{U}$ and $\mathbf{Q}$ are known. Premultiplying with a vector w^T of length $[(s+1)m]$ and moving all known variables to the left side yields

$$r(t) = w^T(\mathbf{Y}(t) - \mathbf{QU}(t)) = w^T\mathbf{R}x(t-s) \tag{3.11}$$

As was described in section 3.2, equation (3.11) will qualify as a residual if the residual is invariant to state variables, i.e.

$$w^T\mathbf{R}x(t-s) = 0 \tag{3.12}$$

Given a vector w that satisfies (3.12) we have a residual generator where the left hand side of (3.11) is the computational form and the right hand side is the internal form.

Residual invariance

Earlier we have assumed it possible to achieve invariance to unmeasured signals, here a method for achieving invariance is presented. If we drop the fault-free no disturbance assumption made in (3.9) the residual generator (3.11) transforms into

$$\begin{aligned} r(t) &= w^T(\mathbf{Y}(t) - \mathbf{QU}(t)) = \\ &= w^T(\mathbf{R}x(t-s) + \mathbf{QF_a}(t) + \mathbf{VF_c}(t) + \mathbf{TN}(t) + \mathbf{S}(t)) \end{aligned} \tag{3.13}$$

where

$\mathbf{F_a}$ is a vector of (unknown) actuator faults

$\mathbf{F_c}$ is a vector of (unknown) component faults

$\mathbf{N}$ is a vector of (unknown) disturbances

$\mathbf{S}$ is a vector of (unknown) sensor faults

$\mathbf{T}$ relates to $\mathbf{N}(t)$ as $\mathbf{Q}$ relates to $\mathbf{U}$(t). $\mathbf{V}$ relates to $\mathbf{F_c}(t)$ as $\mathbf{Q}$ relates to $\mathbf{U}$(t). It can be seen that $\mathbf{T}$ has the same structure as $\mathbf{Q}$ with B changed to E and $D = 0$.

If we also want the residual (3.13) to be insensitive to the unknown disturbances or actuator faults we add the additional constraint:

$$w^T\left[\mathbf{T}\ \tilde{\mathbf{Q}}\tilde{\mathbf{V}}\right] = [0\ 0\ 0] \tag{3.14}$$

where $\tilde{\mathbf{Q}}$ are the $\mathbf{Q}$ matrix where only the columns in the B and D matrices corresponding to inputs to decouple are left and $\tilde{\mathbf{V}}$ are the $\mathbf{V}$ matrix where only the columns in the H matrix corresponding to component faults to decouple are left.

If we want the residual to be insensitive to sensor faults we make sure that all w_i that appears in front of the sensor whose fault we wish to make the residual insensitive to are set to 0. This implies (s+1) zeros per sensor fault.

Diagnostic limits

Of course it is not possible to make the residual insensitive to an arbitrary number of disturbances and faults. We will now derive some of those limits.

What conditions must be fulfilled to make it possible to find a w that satisfies (3.12), (3.14) and then how many actuator/sensor faults are possible to decouple from the residual.

We first note that if we see disturbance as an (unknown) input we only need to consider actuator and sensor fault decoupling. Further we assume that the number of inputs, $n_u \leq n$ where n is the system order and n_u includes the number of disturbances acting upon the system.

Denote the number of actuator faults and disturbances we want to decouple by s_u and the number of sensor faults by s_y. We note that

- To decouple the state influence on the residual, i.e. fulfill (3.12), we have to impose n constraints on w.
- When decoupling s_y outputs we set $s_y(s+1)$ elements in $w = 0$.
- To decouple s_u actuator faults we impose $s_u(s+1)$ if $D \neq 0$ and $s_u s$ if $D = 0$ constraints on w. The special case when $D = 0$ is easy to see when the last column in $\tilde{\mathbf{Q}}$ then becomes all zero.

In [Gertler, 1991] s is chosen as $s = n$ if $D \neq 0$ and $s \geq n - s_u$ if $D = 0$. Summarizing and assuming $s = n$ if $D \neq 0$ and $s = n - s_u$ if $D = 0$, we can see that the number of constraints on w are:

$$n_c = \begin{cases} n + (s_u + s_y)(n+1) & \text{, if } D \neq 0 \\ n + s_u(n - s_u) + s_y(n - s_u + 1) & \text{, if } D = 0 \end{cases}$$

The w vector have as we earlier noted $[(s+1)m]$ elements and to ensure a solution other than the trivial $w = 0$ we need $(s+1)m > n_c$, i.e. an under determined equation system.

That is if $D \neq 0$

$$\begin{aligned} & (n+1)m > n + (s_u + s_y)(n+1) \\ \Rightarrow \quad & s_u + s_y < m - \frac{n}{n+1} = m - 1 + \frac{1}{n+1} \end{aligned}$$

We also know that $n > 0 \Rightarrow \frac{1}{n+1} > 0$, which yields the upper limit on how many faults/disturbances we can decouple.

$$s_u + s_y = m - 1$$

If $D = 0$ we get

$$\begin{aligned} & (n - s_u + 1)m > n + s_u(n - s_u) + s_y(n - s_u + 1) = \\ = \quad & (s_u + s_y)(n - s_u + 1) + n - s_u \\ \Rightarrow \quad & s_u + s_y < m - \frac{n - s_u}{n - s_u + 1} = m - 1 + \frac{1}{n + 1 - s_u} \end{aligned}$$

We also know from the discussion above concerning an upper limit on number of inputs n_u that $n \geq n_u \geq s_u \Rightarrow \frac{1}{n+1-s_u} > 0$ which yields the upper limit on how many faults/disturbances we can decouple even here gets

$$s_u + s_y = m - 1$$

Design example

The example system is a linearized mean-value model of an SI-engine. The model has two states, n the crankshaft rotational speed and p_{man}, the pressure in the intake manifold. One structured disturbance is acting upon the system, the road-load, i.e. up/down-hill etc. There are 3 sensors measuring

- Crankshaft revolution speed (rpm)
- Intake manifold pressure p_{man} (kPa)
- Air flow past the throttel $\dot{m}_{at}$ (kg/h)

The process also consists of two actuators

- Throttle actuator
- Fuel injector

We are here considering sensor faults on all sensors, actuator faults on both actuators, and a component fault as leakage in the intake manifold. The linearized model are:

$$\begin{aligned}
\Delta x(t+1) &= A\Delta x(t) + B\Delta u(t) + Ed(t) + H_1 \begin{pmatrix} f_{a1}(t) \\ f_{a2}(t) \\ f_{c1}(t) \end{pmatrix} \\
\Delta y(t) &= C\Delta x(t) + D\Delta u(t) + H_2 \begin{pmatrix} f_{a1}(t) \\ f_{s1}(t) \\ f_{s2}(t) \\ f_{s3}(t) \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
A &= \begin{pmatrix} -1.6688 & 4.1250 \\ -0.2926 & -15.8177 \end{pmatrix} & B &= \begin{pmatrix} 0 & 410.3077 \\ 55.6064 & 0 \end{pmatrix} \\
E &= \begin{pmatrix} -23.3822 \\ 0 \end{pmatrix} & H_1 &= \begin{pmatrix} 0 & 410.3077 & 0 \\ 55.6064 & 0 & 5.3471 \end{pmatrix} \\
C &= \begin{pmatrix} 1.0000 & 0 \\ 0 & 1.0000 \\ 0 & -0.6655 \end{pmatrix} & D &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 10.3995 & 0 \end{pmatrix} \\
H_2 &= \begin{pmatrix} 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 10.3995 & 0 & 0 & 1.0000 \end{pmatrix}
\end{aligned}$$

	a_1	a_2	s_1	s_2	s_3	M_{load}	c_1
r_1	0	0	1	1	1	0	1
r_2	1	0	1	1	1	0	1
r_3	1	0	0	1	1	0	**0**
r_4	1	0	1	0	1	0	1
r_5	1	0	1	1	0	0	1
r_6	1	0	**0**	1	1	0	0

Table 3.2: Coding set

	a_1	s_2	s_3	c_1
r_1	0	1	1	1
r_4	1	0	1	1
r_5	1	1	0	1
r_6	1	1	1	0

Table 3.3: Reduced coding set

where $\Delta x = \begin{pmatrix} \Delta n \\ \Delta p_{man} \end{pmatrix}$, $\Delta u = \begin{pmatrix} \Delta\alpha \\ \Delta\dot{m}_{fi} \end{pmatrix}$, $d = M_{load}$,
$\begin{pmatrix} f_{a1} \\ f_{a2} \end{pmatrix} = \begin{pmatrix} \text{Throttle actuator fault} \\ \text{Fuel injector fault} \end{pmatrix}$, $f_{c1} =$ Manifold leak and
$\begin{pmatrix} f_{s1} \\ f_{s2} \\ f_{s3} \end{pmatrix} = \begin{pmatrix} \text{rpm-sensor fault} \\ p_{man}\text{-sensor fault} \\ \dot{m}_{at}\text{-sensor fault} \end{pmatrix}$

To isolate all 6 different type of faults we can design a residual vector of dimension 6, each component independent of one fault each. All components should also be independent of the disturbance d. This is however not possible for this model, this can easily be seen as the disturbance d enters the system dynamics in the same way as faults in the $\dot{m}_{fi}$-sensor, f_{a2}. This means that any component decoupling disturbance, automatically decouples any faults in the $\dot{m}_{fi}$-sensor. This is seen in the resulting coding set in table 3.2, the second column corresponding to a_2 is all zero. We also note that the c_1 and the s_1 columns are equal indicating that this scheme is not able to distinguish between the two faults. Usually the rpm-sensor s_1 is very reliable, therefore can the fault code for these two columns be assumed indicating a manifold leakage.

As we now only have 4 faults left to diagnose, we can reduce the dimension to 4. Removing the columns for a_2, s_1 and M_{load} and residuals r_2 and r_3 results in the reduced coding set in table 3.3 that is a strongly isolating coding set.

The time window, s, is chosen as described earlier ($D \neq 0$) to $s = n = 2$. Matlab code to generate the first residual component r_1, insensitive to load

disturbances and faults in the rpm-sensor, can be written as

```
Q = [[D;C*B;C*A*B], [zeros(size(D));D;C*B], [zeros(size([D;C*B]));D]];
T = [[zeros(3,1);C*E;C*A*E], [zeros(3,1);zeros(3,1);C*E],
     [zeros(size([zeros(3,1);C*E]));zeros(3,1)]];
R = [C;C*A;C*A*A];
%%%%% Decoupling, d1 + actuator1 faults
Qtilde = [[D(:,1);C*B(:,1);C*A*B(:,1)], [zeros(size(D(:,1)));D(:,1);C*B(:,1)],
          [zeros(size([D(:,1);C*B(:,1)]));D(:,1)]];
Z = zeros(7,9);
Z(1:2,:) = R';
Z(3:4,:) = T(:,1:2)';
Z(5:7,:) = Qtilde(:,1:3)';
w_temp = Z(:,[1:4,6:7,9])\(-Z(:,5)-5*Z(:,8));
w1 = [w_temp(1:4);1;w_temp(5:6);5;w_temp(7)];
```

Components r_4, r_5 and r_6 are generated with similar code. This residual generator is now simulated in Figure 3.9. Note how the step in load ($\approx$ uphill) affects the speed at $t = 2$. The lowest plot, the α-plot, illustrates how the assumed throttle angle is 28° but at $t = 5$ a 3° fault happens as indicated by the dotted line, also note how this (unwanted) increase in throttle angle affects the crank-shaft speed. Figure 3.10 shows the corresponding residuals. As expected (column 1 in table

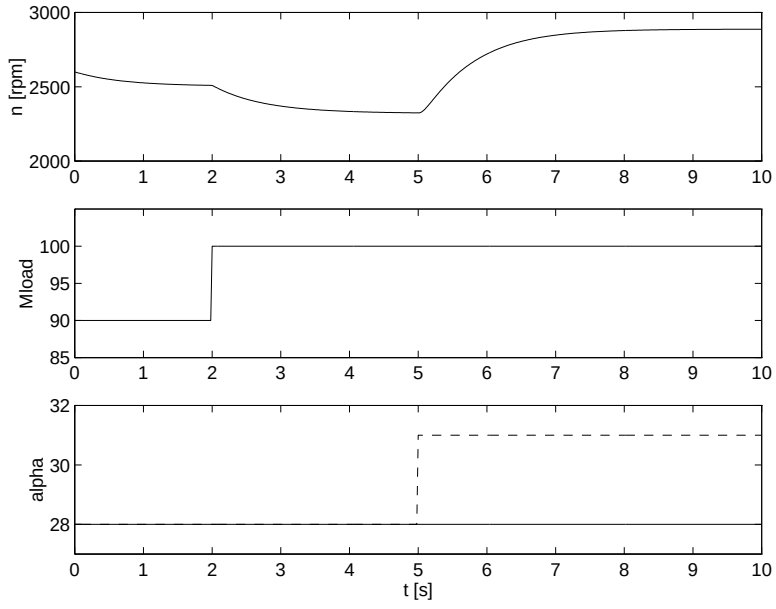

Figure 3.9: Linear throttle fault simulation

3.3) r_4, r_5 and r_6 fires at $t = 5$ while r_1 does not. Note the invariance to the M_{load} step at $t = 2$.

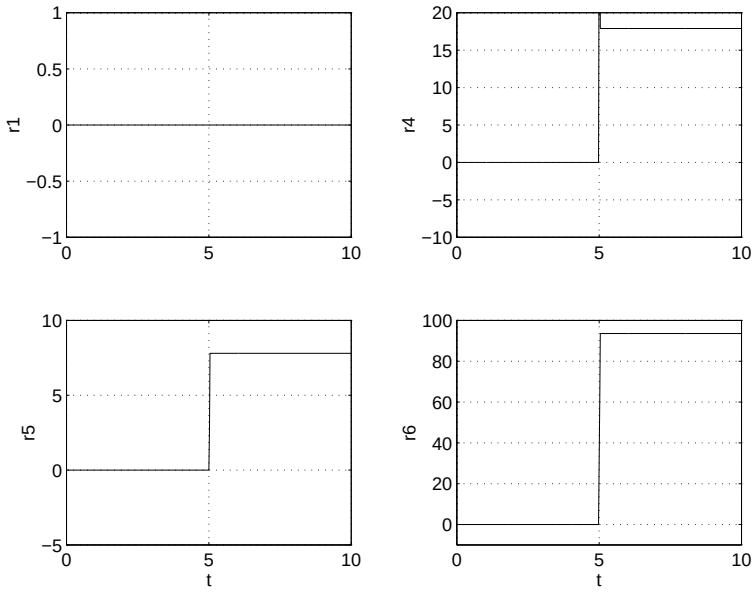

Figure 3.10: Residuals of linear throttle fault simulation

Bibliography

[A.S.—Willsky and Jones, 1974] A.S.—Willsky and H.L. Jones. A generalized likelihood ratio approach to state estimation in linear systems subject to abrupt changes. Proc. IEEE Conf. on Decision and Control, pages 846–853, Pheonix, USA, November 1974.

[Chen and Patton, 1994] J. Chen and R.J. Patton. A re-examination of fault detectability and isolability in linear dynamic systems. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 567–573, Espoo, Finland, 1994.

[Chow and Willsky, 1984] E.Y. Chow and A.S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Trans. on Automatic Control*, 29(7):603–614, 1984.

[Ding and Frank, 1991] X. Ding and P.M. Frank. Frequency domain approach and threshold selector for robust model-based fault detection and isolation. IFAC Fault Detection, Supervision and Safety for Technical Processes, pages 271–276, Baden-Baden, Germany, 1991.

[Frank and Ding, 1994] P.M. Frank and X. Ding. Frequency domain approach to optimally robust ersidual generation and avaluation for model-based fault diagnosis. *Automatica*, 30(5):789–804, 1994.

[Frank and Wünnenberg, 1989] P.M. Frank and J. Wünnenberg. *Robust fault diagnosis using unknown input observer schemes*, chapter 3. In Patton et al. [1989], 1989.

[Frank, 1991] P.M. Frank. Enhancement of robustness in observer-based fault detection. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 99–111, Baden-Baden, Germany, 1991.

[Frank, 1993] P.M. Frank. Advances in observer-based fault diagnosis. Proc. TOOLDIAG'93, pages 817–836, Toulouse, France, 1993. CERT.

[Frank, 1994] P.M. Frank. Application of fuzzy logic to process supervision and fault diagnosis. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 507–514, Espoo, Finland, 1994.

[Gertler and Costin, 1994] J. Gertler and M. Costin. Model-based diagnosis of automotive engines. IFAC Fault Detection, Supervision and Safety for Technical Processes, pages 393–402, Espoo, Finland, 1994.

[Gertler, 1991] J. Gertler. Analytical redundancy methods in fault detection and isolation-survey and synthesis. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 9–21, Baden-Baden, Germany, 1991.

[Hsu *et al.*, 1995] P.L. Hsu, K.L. Lin, and L.C. Shen. Diagnosis of multiple sensor and actuator failures in automotive engines. *IEEE transactions on vehicular technology*, 44(4):779–789, November 1995.

[Isermann, 1989] R. Isermann. *Process fault diagnosis based on dynamic models and parameter estimation methods*, chapter 7. In Patton et al. [1989], 1989.

[Isermann, 1991] R. Isermann. Fault diagnosis of machines via parameter estimation and knowledge processing. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 43–55, Baden-Baden, Germany, 1991.

[Krishnaswami and Rizzoni, 1994] V. Krishnaswami and G. Rizzoni. Non-linear parity equation residual generation for fault detection and isolation. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 305–310, Espoo, Finland, 1994.

[Neumann, 1991] D. Neumann. Fault diagnosis of machine-tools by estimation of signal spectra. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 147–152, Baden-Baden, Germany, 1991.

[Olin and Rizzoni, 1991] P.M. Olin and G. Rizzoni. Robust fault detection. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 259–264, Baden-Baden, Germany, 1991.

[Patton and Chen, 1991] R.J. Patton and J. Chen. A review of parity space approaches to fault diagnosis. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 65–81, Baden-Baden, Germany, 1991.

[Patton and Kangethe, 1989] R.J. Patton and S.M. Kangethe. *Robust fault diagnosis using eigenstructure assignment of observers*, chapter 4. In Patton et al. [1989], 1989.

[Patton *et al.*, 1989] R.J. Patton, P. Frank, and R. Clark, editors. *Fault diagnosis in Dynamic systems.* Systems and Control Engineering. Prentice Hall, 1989.

[Patton, 1994] R.J. Patton. Robust model-based fault diagnosis:the state of the art. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 1–24, Espoo, Finland, 1994.

[Walker, 1989] B.K. Walker. *Fault detection threshold determination using Markov theory*, chapter 14. In Patton et al. [1989], 1989.

Chapter 4

Statistical change detection

In this section we will review statistical change detection methods. We will discuss residual generation, a key theme in fault detection, overview some algorithms and relevant performance measures. An application example is given as well, for which two algorithms are explained in detail. The subject of *diagnosis* will not be treated explicitely. However, a common property of this approach is that it provides an estimate of the change in the system, which may facilitate in the diagnosis step.

There are a number of survey papers in this field, but state of the art is the book [Basseville and Nikiforov, 1993]. As a research field, statistical change period was at its top in the late seventies and early eighties, but during the last few years the interest has increased again.

4.1 Residual generation

There are basically two different ways to generate residuals, that are both based on a model:

- Completely known model. In this case, we essentially run the model in parallel with the system and compute the residual as the difference. In a statistical setting, this is achieved via state estimation by a Kalman filter.
- Parametrized model. The parameters in the model, assumed to be of a linear regression structure, are estimated recursively. The output from the estimated model is then compared to the system. The filter that does this operation is the Recursive Least Squares (RLS) filter.

To get a good understanding of the Kalman and RLS filter a complete course in model-based signal processing would be necessary. However, for understanding statistical change detection, we only need to know the following property: The Kalman and RLS filter take the measured signals and transform them to a sequence of residuals, as illustrated in Figure 4.1. From a change detection point

of view, it does not matter which case we have and the modeling phase can be seen as a standard task.

Figure 4.1: A whitening filter takes the observed input u_t and output y_t and transforms them to a sequence of residuals ε_t.

In a perfect world, the residuals would be zero before a change and non-zero afterwards. Since measurement noise and process disturbances are fundamental problems in the statistical approach to change detection, the actual value of the residuals cannot be predicted. Instead we have to rely on their behavior in average.

If there is no change in the system, and the model is correct, then the residuals are so called white noise, that is a sequence of independent stochastic variables with zero mean and known variance.

After the change either the mean or variance or both changes, that is, the residuals become "large" in some sense. The main problem in statistical change detection is to decide what "large" is.

4.2 Performance measures

A general *on-line* statistical change detector can be seen as a device that takes a sequence of observed variables and at each time makes a binary decision if the system has undergone a change. The following measures are critical:

- Mean time between false alarms (MTFA). How often do we get alarms when the system has not changed?
- Mean time to detection (MTD). How long do we have to wait after a change until we get the alarm?
- Average run length function, ARL(θ). A function that generalizes MTFA and MTD. How long does it take before we get an alarm after a change of size θ.

In practical situations, either MTFA or MTD is fixed, and we optimize the choice of method and design parameters to minimize the other one. For instance, in an airborn navigation system the MTFA might be specified to one false alarm per 10^5 flight hours, and we want to get the alarms as fast as possible under these premises.

In *off-line* applications, we have a batch of data and want to find the time instants for system changes as accurately as possible. This is usually called segmentation. Logical performance measures are:

- Estimation accuracy. How accurate can we locate the change times?
- The Minimum Description Length (MDL). How much information is needed to store a given signal?

The latter measure is relevant in data compression and communication areas, where disk space and bandwidth are limited. MDL measures the number of binary digits that are needed to represent the signal and segmentation is one tool for making this small. Many telephone systems are modelling the signal as autoregressive models over a certain segmentation of the signal, and then transmits only the model parameters, change times (in practice fixed to every 50 ms) and the residuals. The receiver can then recover the signal from this information.

4.3 Change detection methods

Algorithmically, all proposed change detectors can be put in one of the following three cathegories.

- Methods using one model.
- Methods using two models.
- Methods using many model.

In the next subsections, these will be briefly described. Let us already here note that the computational complexity of the algorithm is proportional to how many models that are used. Before reviewing these methods, we need a tool for deciding whether a result is significant or not.

Stopping rules

Many change detection problems, among these algorithms in the classes of one-model and two-model approaches below, can be recast into the problem of deciding on the following two hypotheses:

$$\begin{aligned} H_0: &\quad E(s_t) = 0 \\ H_1: &\quad E(s_t) > 0 \end{aligned}$$

This is essentially achieved by low-pass filtering s_t and comparing this value to a threshold. Below are two such low-pass filters listed:

- The Cumulative sum (CUSUM) test of [Page, 1954]:

$$g(t) = \max(g(t-1) + s(t) - \nu, 0), \quad \text{alarm if } g(t) > h$$

  The *drift parameter* ν influences the low-pass effect, and the *threshold* h (and also ν) influences MTFA and MTD.

- The geometric moving average (GMA) test in [Roberts, 1959]

$$g(t) = \lambda g(t-1) + (1-\lambda)s(t), \quad \text{alarm if } g(t) > h$$

  Here the forgetting factor λ is used to tune the low-pass effect and the threshold h to tune MTFA and MTD.

One-model approach

Statistical whiteness tests can be used to test if the residuals are white noise as they should be if there is no change. Among these tests, we mention

- Change in the mean. The residual itself is used and $s_t = \varepsilon_t$.
- Change in variance. The squared residual subtracted by known residual variance λ is used and $s_t = \varepsilon_t^2 - \lambda$.
- Change in correlation. The correlation between the residual and past inputs and/or inputs are used and $s_t = \varepsilon_t y_{t-k}$ or $s_t = \varepsilon_t u_{t-k}$ for some k.

The first approach is the original CUSUM test of [Page, 1954]. The second one is usually labelled just the χ^2 test, since the test statistic is χ^2 distributed, while a variant of the last one is called the local asymptotic approach in [Benveniste *et al.*, 1987].

4.4 Two-model approach

A model based on only recent data is compared to a model based on data from an infinite time horizon, see figure 4.2. By recent data is often meant data from a sliding window, whose size is denoted L below.

If the former model gives smaller residuals,

$$\|\varepsilon_t^1\| > \|\varepsilon_t^2\|,$$

then a change is detected. The problem here is to choose a norm that correspond to a relevant statistical measure. Two norms that have been proposed are:

- The generalized likelihood ratio (GLR), see [Appel and Brandt, 1983].
- The divergence test, see [Basseville and Benveniste, 1983].

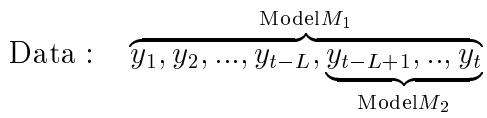

Figure 4.2: The two-model approach. A model (M_2) based on data from a sliding window of size L is compared to a model (M_1) based on all data or a substantially larger sliding window.

Both these criteria provide an s_t to be put in a stopping rule in Section 4.3, for instance the CUSUM test.

The choice of window size L is critical here. On one hand, a large value is needed to get an accurate model in the sliding window and on the other hand a small value is needed to get a short time to detection.

4.5 Multi-model approach

An important property of the Kalman filter and RLS filter is the following observation: If the change time, or set of change times, is known, then the filter can be tailored to this knowledge and give white residuals even after the change(s). Such filters are usually called *matched filters*, because they are matched to specific assumptions on the true system.

The idea in the multi-model approach is to enumerate all conceivable hypotheses about change times and compare the residuals from their matched filters. The filter with the "smallest" residuals wins, and we have an estimate of the change times that usually is quite accurate. The formulation is in a sense off-line, but many proposed algorithms are on-line.

Again, we must decide what is meant by "small" residuals. By just taking the norm of the residuals, we can make it smaller and smaller by increasing the number of hypothesized change times. That is, a penalty on the number of changes must be built-in. The following ones have been proposed:

- The generalized likelihood ratio (GLR) test, see [Willsky and Jones, 1976], and the marginalized likelihood ratio (MLR) test, see [Gustafsson, 1996a].
- The maximum likelihood (ML) approach, see [Gustafsson, 1996b].
- The MDL approach, proposed in [Rissanen, 1978] in a different context.

It is clearly infeasible to apply all possible matched filters to the data. Since there can be a change or no change at each time instant, there are 2^t possible matched filters at time t. Much effort has been spent in developing intelligent search schemes that only keep a constant number of filters at each time.

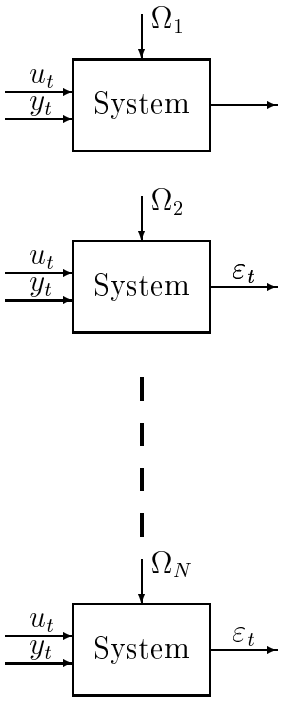

Figure 4.3: A bank of matched filters, each one based on a particular assumption on the set of change times $\Omega = \{k_i\}_{i=1}^{n}$.

4.6 Example: fuel monitoring

The following application illustrates the use of change detection for improving signal quality. The data consist of measurements of instantaneous fuel consumption available from the electronic injection system in a Volvo 850 GLT used as a test car. The raw data are quite noisy and need some kind of filtering before being displayed to the driver at the dashboard. There are two requirements on the filter:

- Good attenuation of noise is necessary in order to being able to tune the accelerator during cruising.
- Good tracking ability. Tests show that fuel consumption very often changes abruptly, especially in city traffic.

These requirements are contradictory for standard linear filters. Figure 4.4 shows the raw data together with a filter implemented by Volvo[1]. Volvo uses a quite fast low-pass filter to get good tracking ability and then quantize the result to a multiple of 0.3 to attenuate some of the noise. However, the quantization introduces a difficulty when trying to minimize fuel consumption manually and the response to changes could be faster.

[1]This is not exactly the same filter as Volvo uses, but the functionality is the same.

Below two of the algorithms are detailed.

Algorithm 1 (The two-sided CUSUM test)

$$\begin{aligned}
\hat{\theta}_t &= \frac{1}{t-t_0}\sum_{k=t_0+1}^{t} y_k \\
\varepsilon_t &= y_t - \hat{\theta}_{t-1} \\
s_t^1 &= \varepsilon_t \\
s_t^2 &= -\varepsilon_t \\
g_t^1 &= \max(g_{t-1}^1 + s_t^1 - \nu, 0) \\
g_t^2 &= \max(g_{t-1}^2 + s_t^2 - \nu, 0) \\
\text{Alarm} &\quad \text{if } g_t^1 > h \text{ or } g_t^2 > h
\end{aligned}$$

After an alarm, reset $g_t^1 = 0$, $g_t^2 = 0$ *and* $t_0 = t$.
Design parameters*:* ν, h.
Output: $\hat{\theta}_t$

Algorithm 2 (Brandt's GLR)

$$\begin{aligned}
\hat{\theta}_t^1 &= \frac{1}{t-t_0}\sum_{k=t_0+1}^{t} y_k \\
\hat{\theta}_t^2 &= \frac{1}{t-L}\sum_{k=L+1}^{t} y_k \\
\hat{\lambda}_t^1 &= \frac{1}{t-t_0}\sum_{k=t_0+1}^{t} (y_k - \hat{\theta}_t^1)^2 \\
\hat{\lambda}_t^2 &= \frac{1}{t-L}\sum_{k=L+1}^{t} (y_k - \hat{\theta}_t^2)^2 \\
\varepsilon_t^1 &= y_t - \hat{\theta}_{t-1}^1 \\
\varepsilon_t^2 &= y_t - \hat{\theta}_{t-1}^2 \\
s_t &= \log\left(\frac{\hat{\lambda}_t^1}{\hat{\lambda}_t^2}\right) + \frac{\left(\varepsilon_t^1\right)^2}{\hat{\lambda}_t^1} - \frac{\left(\varepsilon_t^2\right)^2}{\hat{\lambda}_t^2} \\
g_t &= \max(g_{t-1} + s_t - \nu, 0) \\
\text{Alarm} &\quad \text{if } g_t > h
\end{aligned}$$

After an alarm, reset $g_t = 0$ *and* $t_0 = t$.
Design parameters*:* ν, h, L.
Output: $\hat{\theta}_t^1$

These algorithms are only capable to follow abrupt changes. For incipient changes, the algorithm will give an alarm only after the total change is large or after a long time. In both algorithms, it is advisable to include data forgetting in the parameter estimation to allow for a slow drift in the mean of the signal.

Table 4.1 shows how some change detection algorithms perform on this signal.

Method	Design parameters	$\hat{n}$	MDL	kFlops
RLS	$\lambda = 0.9$, quant $= 0.3$	–	–	19
CUSUM	$h = 3,\ \nu = 2$	14	8.39	20
Brandt's GLR	$h = 20,\ \nu = 3,\ L = 3$	13	8.40	60
Divergence	$h = 20,\ \nu = 3,\ L = 3$	14	8.66	63
ML	$\sigma^2 = 3$	14	8.02	256

Table 4.1: Simulation result for fuel consumption

Figure 4.5 shows the result using the recursive ML detector as a non-linear filter. A changing mean model was used with 5 parallel filters. Compared to the existing filter, the tracking ability has improved slightly and, more importantly, the accuracy gets better and better in segments with constant fuel consumption.

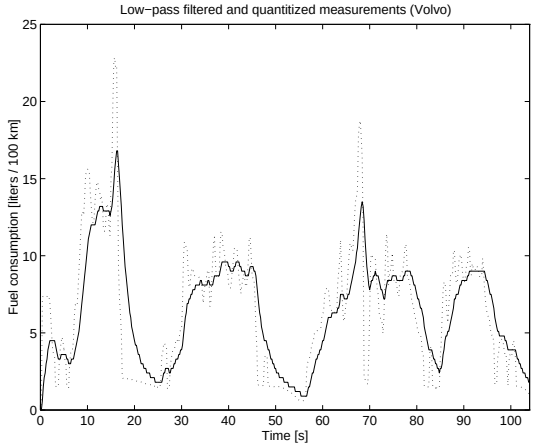

Figure 4.4: Measurements of fuel consumption, dotted, and Volvo's proposed filter, solid.

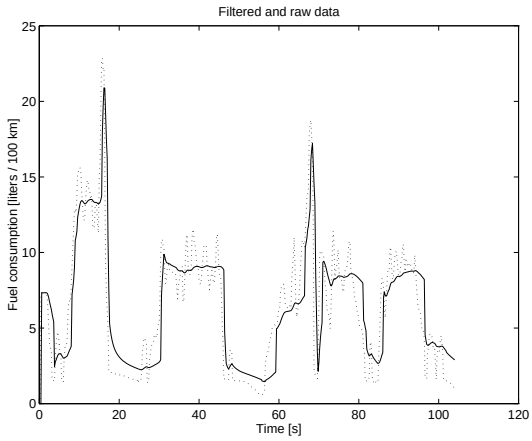

Figure 4.5: Filtering with recursive ML detection

Bibliography

[Appel and Brandt, 1983] U. Appel and A.V. Brandt. Adaptive sequential segmentation of piecewise stationary time series. *Information Sciences*, 29(1):27–56, 1983.

[Basseville and Benveniste, 1983] M. Basseville and A. Benveniste. Design and comparative study of some sequential jump detection algorithms for digital signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31:521–535, 1983.

[Basseville and Nikiforov, 1993] M. Basseville and I.V. Nikiforov. *Detection of abrupt changes: theory and application*. Information and system science series. Prentice Hall, Englewood Cliffs, NJ., 1993.

[Benveniste *et al.*, 1987] A. Benveniste, M. Basseville, and B.V. Moustakides. The asymptotic local approach to change detection and model validation. *IEEE Transactions on Automatic Control*, 32:583–592, 1987.

[Gustafsson, 1996a] F. Gustafsson. The marginalized likelihood ratio test for detecting abrupt changes. *IEEE Trans. on Automatic Control*, 41(1):66–78, 1996.

[Gustafsson, 1996b] F. Gustafsson. Optimal segmentation of linear regression parameters. *Accepted for publication in IEEE Trans. on Signal Processing and available at* `http://www.control.isy.liu.se/ fredrik/journal`, 1996.

[Page, 1954] E.S Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Roberts, 1959] S.W. Roberts. Control charts based on geometric moving averages. *Technometrics*, 8:411–430, 1959.

[Willsky and Jones, 1976] A.S. Willsky and H.L. Jones. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Transactions on Automatic Control*, pages 108–112, 1976.

Chapter 5

Discrete model-based diagnosis

This chapter starts out with an introduction to diagnostic reasoning before focusing on model based diagnosis.

5.1 Introduction to diagnostic reasoning

Reasoning in this context means reasoning about a physical system. If we let the physical system be our device then diagnostic reasoning can be viewed as reasoning about what is wrong with that device.
There are several approaches to diagnostic reasoning. The simplest and most straightforward is *fault-based* or *rule-based* reasoning. Fault-based reasoning imitates a repair-manual to maintain heuristic knowledge about the device. This is used to guide the diagnosis. The problem is that this approach can only deal with problems known in the "manual".
Another approach taken by Poole [1988] and Reiter [1987] is *default* reasoning. Default reasoning provides the ability to reason with incomplete information by allowing assumptions to be made based on default rules when actual information is missing. While the ability to make assumptions is the strength of this approach it is also it's weakness. All it takes is one wrong assumption for this approach to fail.
The *case-based* reasoning approach attempts to use knowledge about previous cases to guide the diagnosis. This may work well in cases where the same or very simlilar cases have been encountered before but when the current case is new there are problems. Usually a diagnosis is based on the most similar previously encountered case. To evaluate how close two cases are usually involves some heuristic measurement. Unless this measurement is very good the wrong previous case may be choosen. Also, even if the closest previous case is choosen the actual diagnosis may be completely different.
The last approach accounted for here is *model-based* diagnosis. This is the approach taken by Colsole and Torasso [1991], de Kleer and Williams [1987, 1989]

and Derssler and Struss[1996]. All of these are based on the same idea. A *model* of the correct device is used to compare the observed behaviour of the device to that predicted by the model. Differences between predicted and observed behaviour are symptoms of one or more faults. Diagnosis is done by finding the changes in the model that explains the observed behaviour of the device. Those changes are the diagnosis. The problem with model-based diagnosis is combinatoric explosion, if there are n things that can go wrong with the device there are 2^n possible diagnoses. Finding a diagnosis is a search problem among these 2^n possible solutions. Needless to say some efficient way to find the correct diagnosis is needed.

5.2 Model-based diagnosis

The basic idea of model-based diagnosis is to compare certain measured values from the device to be diagnosed with the values predicted by a model of that device. Any differences between measured and predicted values are called *discrepancies*[1]. The diagnosis task is to identify the fault(s) causing the discrepancy.
The remainder of this section is a more detailed description of model-based diagnosis based on Console and Torrasso's framework [1991].

Some basic definitions

To use Console and Torasso's framework some basic definitions are needed, first of all a device to be diagnosed.

Definition 1 *D is the device to be diagnosed.*

A device is built out of *components*.

Definition 2 *Let $COMP = \{c_1, ..., c_n\}$ be the set of components of D.*

Console and Torasso's framework uses *behaviour-modes*[2] to describe and identify faulty components. Each component c_i is characterised by a set of behaviour-modes.

Definition 3 *For each component c_i in $COMP$ there is a set of behaviour-modes $BM_i = \{correct_i, fault_{i_1}, ..., fault_{i_m}\}$.*

[1] Discrepancies are also called symptoms by some authors. In this text both discrepancies and symptoms will be used.

[2] A behaviour-mode corresponds to what is called fault-mode by some authors.

For each component there is one *correct* behaviour and a number of *faulty* behaviours. The behaviour of the device D can be represented as the consequence of the behaviour-modes of its components.
As suggested by de Kleer and Williams [1989] a behavior-mode for unknown faults could (and should?) be added to cover possible unforseen behaviour-modes. A revised definition would look something like this.

Definition 4 *For each component c_i in $COMP$ there is a set of behaviour-modes $BM_i = \{correct_i, unknown_i, fault_{i_1}, ..., fault_{i_m}\}$.*

To assist in the definition of the model later on two sets are defined, *abducible* and *non-abducible* symbols.
The union of all sets of behaviour-modes is called the set of abducible symbols.

Definition 5 *The set of abducible symbols $ABDSYM = BM_1 \cup ... \cup BM_n$.*

Every abducible symbol corresponds to a behaviour-mode for one of the components in the device D. The following definition helps to express the fact that a certain component is in a certain behaviour-mode.

Definition 6 *Let α be an abducible symbol (behaviour-mode) for component c_i. Component c_i is in behaviour-mode α is written $\alpha(c_i)$.*

For each behaviour-mode (abducible symbol) there is one or more consequences.

Definition 7 *For each behaviour-mode α there is a set $CONS_\alpha = \{consequence_1, ..., consequence_m\}$.*

The union of all sets of consequences defines a set of non-abducible symbols.

Definition 8 *The set of non-abducible symbols $NONABDSYM = CONS_{\alpha_1} \cup ... \cup CONS_{\alpha_n}$.*

The model and system description

A model of the device D is built from the abducible and non-abducible symbols expressed as *Horn clauses*[3].

Definition 9 *Let γ be a Horn clause built from symbols $\beta \in ABDSYM \cup NONABSYM$. Then the model for the device D is the set of Horn clauses $MODEL = \{\gamma_1, ..., \gamma_k\}$ describeing the behaviour of the device.*

[3]A Horn clause is a disjunction of negated litterals and at most one positive litteral $\neg l_1 \vee ... \vee \neg l_n \vee l$.

The main reason for restricting the model to Horn clauses is efficiency. It is possible to test satisfiability of a set of Horn clauses in linear time [Dowling and Gallier, 1984]. Given a model and it's components a system description is defined like this.

Definition 10 *Given a model (MODEL) and components (COMP) for a device D a system description for the device D is a pair*
$SD = \langle MODEL, COMP \rangle$.

The diagnostics problem

To define a diagnostics problem contextual data and observations have to be added. Contextual data and observations are expressed as atoms.

Definition 11 *The atom $f(a)$ expresses that a is the value of parameter f.*

Now contextual data can be defined as a set of atoms representing the inputs to the device D. Console and Torrasso [1991] defines contextual data as a set of parameters providing information about the specific case under examination.

Definition 12 *Let $f_1, ..., f_n$ be inputs to the device D and $a_1, ..., a_n$ their values. The contextual data CTX is the set of atoms*
$CTX = \{f_1(a_1), ..., f_n(a_n)\}$.

Contextual data need not be accounted for by a diagnosis but is used to predict the behaviour of the device.
Observations can be defined as a set of atoms representing the outputs of the device D. The definition is very much like the one for contextual data.

Definition 13 *Let $f_1, ..., f_n$ be outputs of the device D and $a_1, ..., a_n$ their values. The observations OBS is the set of atoms $OBS = \{f_1(a_1), ..., f_n(a_n)\}$.*

The important difference from contextual data is that the observations have to be accounted for by a diagnosis.
The following constraint makes sure a parameter can't have more than one value and that it can't be both an input and an output.

Constraint 1 *Each parameter can only appear once in $CTX \cup OBS$.*

Now a diagnostics problem can be defined as a triple consisting of a system description, contextual data and observations.

Definition 14 *Let SD be the system description for device D, CTX be the contextual data and OBS be the observations. Then the diagnostics problem DP is defined as a triple $DP = \langle SD, CTX, OBS \rangle$.*

The solution to the diagnostics problem

Solving the diagnostics problem for a device is equivalent to identifying the behaviour-modes of it's components that "explain" the observations. To do this two notions of explanation are defined. First a weak notion of explanation.

Definition 15 *Explanation as consistency: A diagnosis explains an observation m if does not contradict m.*

Then there is the strong notion of explanation.

Definition 16 *Explanation as covering: A diagnosis explains an observation m if it directly supports m.*

These two defintions are the basis for two classical approaches to solving the diagnosis problem. The consistensy based approach relies on a weak notion of explanation. In this approach it is sufficient that the assigned behaviour-modes are consistent with the observations. The abduction based approach relies on a strong notion of explanation and requires that the observations are covered by the assigned behaviour modes.
Console and Torrasso [1991] uses an *abduction with consistensy constraints* approach. This is a combination of the two classical approaches that relies both on strong and weak notion of explanation.
Two sets Ψ^+ and Ψ^- are defined.

Definition 17 *The set $\Psi^+ \subseteq OBS$ is the set of observation atoms $(f_i(a_i))$ that have to be covered by a diagnosis.*

Definition 18 *The set $\Psi^- = \{\neg f(x) \mid f(y) \in OBS, x \neq y\}$ is the set of negated atoms that conflict with the observations in Ψ^+.*

The set Ψ^- can be interpreted as the set of consistency constraints that the solution must satisfy[4].
Now an abduction problem can be defined as follows.

Definition 19 *Given a diagnostic problem $DP = \langle SD, CTX, OBS\rangle$ the corresponding abduction problem AP is a triple*
$AP = \langle SD, CTX, \langle \Psi^-, \Psi^+\rangle\rangle$.

Solving diagnostic problem DP for a device D is equivalent to identifying the behaviour modes of the components in D. Thus the solution is an assignment that assigns exactly one behaviour mode to each component.

[4]This means that ruling out a value a for a parameter f is the same as putting $\neg f(a)$ in Ψ^-. Then the parameter f can't assume the value a without breaking the consistency constraints.

Definition 20 *Given a system description* $SD = \langle MODEL, COMP \rangle$ *and the set of abducible symbols* $ABDSYM$ *in* $MODEL$. *An assigment* W *for* $COMP$ *is a set of abducible atoms* (α) *such that for each component* $c \in COMP$ W *contains exactly one element of the form* $\alpha(c)$.

A solution (explanation) to an abduction problem is an assignment that cover Ψ^+ and is consistent with Ψ^-.

Definition 21 *Given an abduction problem*
$AP = \langle \langle MODEL, COMP \rangle, CTX, \langle \Psi^+, \Psi^- \rangle \rangle$.
An assignment W *is an explanation for* AP *iff*

1. W *covers* Ψ^+, *that is, for each* $m \in \Psi^+$, *we have that* $MODEL \cup CTX \cup W \vdash m$.
2. W *is consistent with* Ψ^-, *that is,* $MODEL \cup CTX \cup W \cup \Psi^-$ *is consistent.*

5.3 Finding a solution

The previous section clearly defines the solution to the abduction problem but does not say anything about how to find that solution. Finding a solution means finding an assignment W that satisfies the conditions in Definition 21. The reminder of this section presents algorithms for finding an assignment.

Simple algorithm

First lets take a closer look at a very simple algorithm. The algorithm is based on pure intuition and simply goes through all possible assignments, testing them and filtering out those that are valid assignments.

Algorithm 1
For *each possible assignment* W.
  If W *covers* Ψ^+ *and* W *is consistent with* Ψ^- **Then**
    W *is a solution.*
End for.

This algorithm will find every possible assignment W that satisfies Definition 21 but does this at the cost of testing all possible assignments. In the simplest possible case with n components that each have two possible behaviour-modes[5] this algorithm has to test 2^n assignments. For very small problems this approach may be possible but as n grows a more effective algorithm is needed.

[5] Correct or faulty.

Fault probabilites

In the simple algorithm no knowledge about the components is assumed. For all we know every assignment is equally likely to be a valid diagnosis and we have no way of choosing one before the other. To improve on the simple algorithm more information about the components is needed. This enables us to make intelligent choices when selecting assignments for testing.
Let us assume that we have some knowledge about the components in the shape of fault-frequencies and that we are only interested in finding the most likely assignment. This next algorithm will do exactly that. For simplicity all possible assignments are assumed to be on a list sorted by fault-frequency with the assignment with the highest fault-frequency first. The algorithm goes through the list untill an assignment is found that fullfills the conditions in Definition 21.

Algorithm 2
Let LW be a list of all possible assignments W sorted by fault-frequency.
Let $Found$ be a boolean variable initiated to $FALSE$.
Let W be a variable initiated to the first element on LW.
While *not $Found$*
  If *W covers Ψ^+ and W is consistent with Ψ^-* **Then**
    W is a solution set $Found = TRUE$
  Else
    W is not a solution set W to be the next element in LW
End While

This algorithm may appear to be better then the simple algorithm but this is not always true, if the only valid assignment is the last one on the list they are just as bad. This is the worst case however, in the average case this algorithm will be better.
Still there are some apperent flaws to this algorithm. It starts testing assignments in the same order every time without considering any information about the specific case.

Dependencies

In the two previous algorithms we haven't considered any information given by the observations. Let us assume that all observations are atoms. Then a *symptom* can be defined [de Kleer and Williams, 1987] but first the definition of an atom (Definition 11) need to be expanded.

Definition 22 *If f is a parameter and a is a value then:*

- *$f(a)$: f has the value a.*

- $f_m(a)$: *f has been measured to value a.*
- $f_p(a)$: *f has been predicted to value a.*

Now a symptom is a measured parameter value that significantly deviates from the predicted parameter value.

Definition 23 *A symptom s_f is a pair $s_f = \langle f_m(a), f_p(b) \rangle$ where $a \neq b$.*

Symptoms can be used to narrow down the search for a valid assignment. If a parameter f is measured from a component c then the correct value of parameter f *depends* on the correct operation of component c. Normally the component c depends on other components $(c_1, ..., c_n)$ which depends on other components and so on. Now when a symptom for parameter f is found we know that something has to be wrong with the components it depends on. Using symptoms and dependencies a more efficient algorithm[6] can be built.
This algorithm assumes that a symptom for a parameter f has been observed. Since f depends on the correct operation of all the components in DEP_f thus the faulty component(s) we are looking for must be in this set. All possible combinations of components in DEP_f are in $DEPSET_f$. Assume that we want to find the valid assignment with the fewest number of faulty components. To do this $DEPSET_f$ is sorted in the order of the smallest subset of DEP_f first. Now finding the assinment we want is only a matter of walking through $DEPSET_f$ and for each element DEP build the corresponding assignment W then test if W fullfills Definition 21.

Algorithm 3
Assume that a symptom for the parameter f have been found.
Let DEP_f be a list of all components parameter f is depending on.
Let $DEPSET_f$ be a set of all possible subsets of DEP_f sorted from the smallest subset to the largest.
Let DEP be a variable initialized to the first element in $DEPSET_f$.
Let Found be a boolean variable initiated to TRUE.
While *not Found*
  Let W be an assignment that assigns all components in DEP to be faulty and all other components to be correct.
  If *W covers Ψ^+ and W is consistent with Ψ^-* **Then**
    W is a solution set $Found = TRUE$
  Else
    W is not a solution set DEP to be the next element in $DEPSET_f$
End While

[6]This algoritm is simplified for illustrative purposes, a complete algorithm using dependencies would be GDE [de Kleer and Williams, 1987].

This algorithm only has to test 2^n possible assignments where n is the number of components the parameter f depends on. Since n should be less than the total number of components this algorithm should have a significant advantage over the two previous algorithms. However if the parameter f depends on all the components then this information is useless and this algorithm is just as bad as the two previous.
There is still plenty of room for improvements in this algorithm. One possible improvement is to make use of fault-frequencys like the second algorithm. It should also be able to make use of multiple symptoms like GDE [de Kleer and Williams, 1987].

5.4 Selecting a solution

In many cases there will be more than one explanation to a given abduction problem. Some way of selecting the correct diagnosis is needed.

Minimal diagnosis

This is the approach taken by Console and Torrasso [1991] and Reiter [1987]. The diagnosis with the least faulty components is selected. To be able to compare explanations each explanation is partitioned into two parts. One partition contains the correct modes.

Definition 24 *$correct(W) = \{correct(c) \mid correct(c) \in W\}$ where c is a component in D and correct corresponds to the correct mode of c.*

Another partition contains the faulty components.

Definition 25 *$faulty(W) = W - correct(W)$.*

Now it is possible to compare explanations and define minimal explanation as the explanation with the least faulty components.

Fault Probabilities

This is the approach used in Sherlock [de Kleer and Williams, 1989]. When using fault probabilities each faultmode is assigned a fault probability[7]. To select the correct diagnosis simply select the explanation with the highest probability mass. This can also be used together with Console and Torasso's framework, the only thing that is needed is to calculate the probability mass of the candidates. De Kleer and Williams is a bit more advanced however since they also use the fault probabilites to guide the diagnosis process.

[7] The fault probability can also be thought of as a fault frequency.

5.5 Speeding up model-based diagnosis

Since efficiency is an important issue for any on-board diagnosis system this section is devoted to methods speeding up model-based diagnosis. The number of possible diagnoses grows fast as the model-becomes more advanced. For example if you have a system with n components that can be either correct or faulty[8] the number of possible diagnoses is 2^n. This is a very large number even for quite small n and if we want to add more behavoiur modes for each component the situation is even worse. Since diagnosis can be seen as search thru these possible diagnoses an efficient searchmethod is needed that (hopfully) does not need to search all possibilites to find a diagnosis.

Case based reasoning

Another approach [Portinale *et al.*, 1996] is to use case based reasoning (CBR) to guide the diagnosis process. The basic idea is to use knowledge about previous diagnosis cases to find a solution to the current one. When the current case matches some previously known case this method is very fast, the diagnosis is found immediately. The problems with this approach occurs when there is no exact match for the current case with previous cases. The brave way to approach this problem is to attempt to find the previously known solution that is closest by some measure[9] and, if needed, adapt this to the current case. The problem is that even though the two cases may be close their solutions may be very far from each other. The adaption process may be very extensive to the point where the used case is missguiding the diagnosis process. The safe way to deal with the problem is to fall back on model-based diagnosis and dissguard previously known and maybe misleading cases but then not much is gained from the knowledge about previous cases. One solution would be to attempt to adapt the previous case and abandon the adaption process when it becomes too extensive. The problem is then to find some good way of telling when to abandon the adaption process.

Multiple models

Dressler and Struss [1996] take a different approach to accomplish higher efficiency. Instead of trying to speed up or guide the search for the correct diagnosis they suggest reducing the size of the model. They use a simplified model to guide the search. By observing that some faults can't occur[10] together some faults rule out others the simplified model can be replaced by more accurate ones as the diagnosis becomes more accurate. These models can be arranged in a tree with

[8] That is each component has two faultmodes, correct and faulty.

[9] This is one weakness of this brave approach, there is no "perfect" way to measure how close two cases are.

[10] Also some faults make it impossible to diagnose some other components.

the top node being the most general and most simplified model. Each branch in the tree represents more accuratee models where the leaf models are the most accuratee.
This method has the potential[11] to reduce the number of possible diagnoses and thus make the diagnosis more efficient. This is done at the cost of multiple models.

5.6 Diagnosis in DEDS

Discrete event dynamic systems (DEDS) are systems where all variables can take only discrete values. We restrict ourselves to such systems where each variable belongs to a finite set. Inspired by the mathematical description for continuous systems $\dot{x} = f(x, u)$ we write a DEDS with n states, p inputs and m outputs as

$$x^+ = f(x, u) \tag{5.1}$$

where x^+ (next state), x, u belong to finite sets X^n, X^n, U^p and the transition function is a mapping $f : X^n \times U^p \to X^n$. The expression (5.1) models an explicit behavior whereas $f(x^+, x, u) = 0$ models an implicit behavior. For outputs we write $y = g(x, u)$, $y \in Y^m$.

The variables x, u, x^+ and y are regarded as signals and the models f, g are signal models. This differs, in respect to representation, from a pure event based approach where a value (or symbol) represents a change in the environment. Instead we say that the environment can always be measured and we do not need to remember values on the inputs. Signal based models has a closer connection to physical systems controlled by some sampling device. Pure event based models can be translated to signal based models, see Gunnarsson [1995].

The relations f and g can be described in several different ways, for example as a table. We will use *binary decision diagrams* (BDD)[Bryant, 1986] to store these relations efficiently.

Diagnosis or fault detection is in this environment the same as an observer. We have to model the system (including possible faults) and then find the set of states which corresponds to the available observations. Unmodelled faults can be discovered if they lead to an abnormal behaviour. The set of possible states delivered by the observer will in this case be an empty set.

[11] This is not a general result, it is possible to create "dumb" cases where it is not true. One simple example is using the same model on every level.

Bibliography

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[Console and Torasso, 1991] L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Comput. Intell.*, 7:133–141, 1991.

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

[de Kleer and Williams, 1989] J. de Kleer and B.C. Williams. Diagnosis with behavioural modes. In *Procedings of the 11th international joint conference on artificial intelligence*, 1989.

[Dowling and Gallier, 1984] W.F Dowling and J.H Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. logic programming*, 3:267–284, 1984.

[Dressler and Struss, 1996] O. Dressler and P. Struss. The consistency based approach to automated diagnosis of devices. In Gerhard Brewka, editor, *Principles of knowledge representation*, chapter 8, pages 269–313. CSLI Publications, 1996.

[Gunnarsson, 1995] Johan Gunnarsson. On modeling of discrete event dynamic systems, using symbolic algebraic methods. Technical Report LiU-TEK-LIC-1995:34, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, June 1995.

[Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Portinale *et al.*, 1996] L. Portinale, P. Torraso, C. Ortalda, and A. Girardino. Using case-based reasoning to focus model-based diagnostic problem solving. Torino, Italy, 1996.

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

Chapter 6

Temporal Reasoning

This chapter surveys some of the work on temporal reasoning within artificial intelligence. Section 6.1 introduces reasoning about temporal constraints, that is, various algebras and methods for drawing conclusions from sets of temporal relationships. Important issues here are the ontological choices and the resulting computational consequences. Since the area is still immature and evolving, there is no common theory to present. Hence, in order to give the interested reader some feeling of the technical details, some in-depth examples are provided in the subsequent section. Section 6.3 then briefly discusses formalisms and methods for reasoning about time together with other information which is temporally qualified (reasoning about knowledge and time). Finally, the last two sections briefly discusses some examples of implemented systems and gives some references to further reading.

6.1 Temporal-Constraint Reasoning

Temporal constraint reasoning is an important task both in AI and elsewhere, having relevance to such diverse areas as planning; natural language processing; model-based diagnosis; time serialization in archeology, history and paleontology, *etc.* Isomorphic problems also arise in areas such as spatial reasoning and molecular biology.

In most applications, information about temporal constraints is expressed as collections of relations between time intervals or time points. Typical reasoning tasks include determining the satisfiability of such collections and deducing new relations from those that are known. There are several ontological choices to make, *eg.* should time points or time intervals be used as basic primitives, should time be quantitative or qualitative *etc.*? These choices affect both the modelling power and the computational complexity, as will be discussed later. However, before going into technical details, we consider an example of a temporal-constraint reasoning problem.

Example: A Murder Scenario

Professor Jones has been found shot on the beach near her house. Rumours tell that she was almost sure of having a proof that P≠NP, but had not yet shown it to any of her colleagues. The graduate student Hill is soon to defend his thesis on his newly invented complexity class $\mathrm{NRQP}_{\Sigma}(\star)^{\infty}$, which would unfortunately be of no value were it to be known for certain that P≠NP. Needless to say, Hill is thus one of the prime suspects and inspector Smith is faced with the following facts and observations:

- Professor Jones died between 6 pm and 11 pm, according to the post-mortem.
- Mr Green, who lives close to the beach, is certain that he heard a gunshot at some time in the evening, certainly after the TV news.
- The TV news is from 7.30 pm to 8.00 pm.
- A reliable neighbour of Hill claims Hill arrived at home sometime between 9.15 pm and 9.30 pm.

Furthermore, the following pieces of general information are known:

- It takes between 10 and 20 mins. to walk or run from the place of the crime to the closest parking lot.
- It takes between 45 and 60 mins. to drive from this parking lot to Hill's home.

As a help in solving this crime, Inspector Smith could use temporal-constraint reasoning as a help to rule out suspects. For instance, in order to see if Hill can be ruled out as a suspect based on the temporal information, Smith could proceed as follows. First he has to check that the set of temporal facts and observations above is consistent—if not, then some observation or fact must be wrong. As a next step, further information may be deduced from what is known. For instance, as is shown in Figure 6.1, the fact that Jones died between 6 pm and 11 pm together with the fact that the shot was heard after the TV news makes it possible to narrow down the time of death to the interval between 8 pm and 11 pm. Now, in order to see if Hill could have been at the beach at the time of the crime, Smith adds the hypothesis that Hill did go (or run from the beach) and then drive home. It is known that Smith did arrive home at some time during the interval from 9.15 pm to 9.30 pm so it is possible to calculate backwards to see when he must have left the beach (if he ever was there). This time can only be determined within some upper an lower bound, but interesting question is whether it is possible for this time point to have occurred during the interval of the death, that is, between 8 pm and 11 pm. If not, then inspector

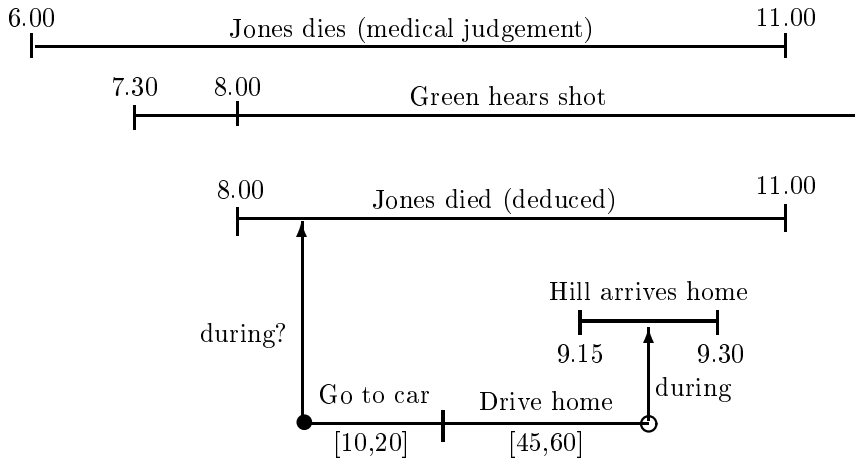

Figure 6.1: Temporal constraints in the murder scenario

Smith can rule out Hill as a suspect. This can be done by testing if the set of known facts and observations together with the hypothesis is inconsistent. If so, then Hill could not have been at the beach at the time of the crime and so is no longer a suspect, while in the other case Smith does not gain any new knowledge so Hill is still a suspect.[1]

It is obvious from this example that there are several different types of temporal entities and different types of relations between these. For instance, the time of death can be viewed as a *time point* which is known to occur at some time *during* an *interval* which starts at 6.00 pm and ends at 11.00 pm, *ie.* we have a time point with unknown occurrence time related to an interval with known metric starting and ending times.

Ontological Choices

As was mentioned in the introduction, we can choose either *time points* or *time intervals* as our primitive temporal entities. Although the question which of these concepts should be regarded as primitive has spurred much philosophical debate, we need not usually be concerned with that question; time points and time intervals can co-exist since a time interval can be viewed as a pair of time points—denoting the starting and ending time respectively of the interval. We will use the symbol t for time points and the symbol I for time intervals (using subscripts whenever necessary). Furthermore, for each interval I, we will denote its starting and ending time points by I^- and I^+ respectively, that is, we have for all intervals I that $I = [I^-, I^+]$.

[1]The alert reader will probably already have observed that Hill will be in need of juridical assistance.

Another distinction to make is between *qualitative* (or relative) time and *quantitative* (or metric) time. If using only qualitative time we may, for instance, say that a time point t_1 is strictly before another time point t_2, denoted $t_1 < t_2$ or that they are the same time point[2], denoted $t_1 = t_2$. However, it is not possible in the first case to express how long before t_2 the time point t_1 occurs. Neither can we say when t_1 and t_2 occur in the second case, although we know they both occur at the same time. Similarly, for two time intervals I_1 and I_2 we may, for instance, know that I_1 occurs strictly before I_2, denoted $I_1\{<\}I_2$ or we may know that I_1 either ends exactly when I_2 starts (*ie.* I_1 *meets* I_2) or that I_1 starts before I_2 starts and ends after I_2 starts but before I_2 ends (*ie.* I_1 *overlaps* I_2), which is denoted $I_1\{m\ o\}I_2$. However, there is no information about the metric durations of the two intervals, or about how long before I_2 starts does I_1 start.

If using quantitative time, on the other hand, we express actual occurrence times of time points, *eg.* $t_1 = 17$, and distances between time points, *eg.* $t_1 - t_2 = 5$. By treating intervals as pairs of time points we may similarly represent durations of intervals, *eg.* $d(I) = I^+ - I^- = 5$, distances between the starting times of intervals, *eg.* ${I_2}^- - {I_1}^- = 7$, *etc.* All these examples represent exact metric information. However, we may also consider allowing uncertain metric information, saying, for instance, that time point t_1 occurs somewhere between the exact times 5 and 7, *ie.* $5 \leq t_1 \leq 7$, or that the duration of interval I_1 is at most 8 time units, *ie.* ${I_1}^+ - {I_1}^- \leq 8$. It is worth noting that if we allow full metric time with uncertainty, then we can express also qualitative relations.

Temporal Algebras

Given some ontological choice of temporal entities and relations, we can also define certain operations in order to get an algebra. The usual operations considered for temporal algebras are *converse*, *intersection* and *composition*. The converse of a relation corresponds to interchanging its arguments, *eg.* the converse of $t_1 < t_2$ is $t_1 > t_2$ (note, not the inverse $t_1 \geq t_2$). As will be discussed below, relations are viewed as sets of basic relations, so the relation $\geq$, for instance, is really an abbreviation for the set $\{= >\}$. The intersection between two relations is simply the intersection between the two sets of basic relations, *eg.* the intersection of $t_1 \leq t_2$ and $t_1 \geq t_2$, which really denotes the intersection between $t_1\{< =\}t_2$ and $t_1\{= >\}t_2$, is $t_1\{=\}t_2$, *ie.* $t_1 = t_2$ in 'sloppy' notation. Finally, composition is essentially a transitivity operation, but is slightly more complex since it has to handle different relations. For instance, the composition of $t_1 \leq t_2$ and $t_2 \leq t_3$ is

[2]Strictly speaking, we should call t_1 and t_2 not time points, but rather time tokens or time-point symbols, since they are only variables denoting time points. The tokens t_1 and t_2 are two distinct symbols which may or may not denote distinct points on the time line. However, trusting the benevolent reader not to be confused by this abuse of language, we will somewhat sloppily refer to time tokens as time points in order to simplify the presentation. An analogous note applies also to time intervals.

$t_1 \leq t_3$, while the composition of $t_1 \leq t_2$ and $t_2 < t_3$ is $t_1 < t_3$.

Qualitative Algebras

The *point algebra* (more precisely the qualitative time point algebra) [Vilain, 1982] uses only *time points* and only *qualitative relations* between these. There are three *basic relation* between time points: *before* ($<$), *equals* ($=$) and *after* ($>$). We further allow *composite relations*, formed as sets of basic relations and denoting the disjunction of its constituent basic relations. For instance, the expression $t_1\{< =\}t_2$ denotes that either $t_1 < t_2$ or $t_1 = t_2$ and is usually abbreviated as $t_1 \leq t_2$. Similarly we can also form the two relations $\{= >\}$ and $\{< >\}$, which are abbreviated as $\geq$ and $\neq$ respectively. The two remaining composite relations are the non-relation $\{< = >\}$ denoting the empty constraint, which is equivalent to having no constraint at all, and the inconsistent relation $\{\}$, which can never be satisfied. All in all, this totals eight different relations, three basic and five composite, corresponding to the eight subsets of $\{< = >\}$. It is worth noting that the two relations $\leq$ and $\neq$ are sufficient to express all the other point-algebra relations and that this fact is exploited in many temporal-reasoning systems in order to simplify the data structures and algorithms. It is also common to view a set of relations as a directed graph with arcs labelled by $\leq$ or $\neq$.

The *interval algebra* [Allen, 1983] (often called the Allen algebra after its inventor[3]) uses time intervals as primitive entities and does not include time points explicitly. While two time points can only be related in three basic ways, two intervals can be related in more complex ways. More precisely the possible relations are: *before* ($<$), *meets* (m), *overlaps* (o), *starts* (s), *during* (d), *finishes* (f) and *equals* ($=$). In addition, there are also the converses of these seven relations, denoted by adding an "i" after the relation symbol, with two exceptions: the inverse of *before* is called *after* and is denoted $>$, and *equals* is the inverse of itself. This totals 13 different basic relations, which are illustrated in Figure 6.2. Just as in the case of the point algebra, we also allow composite relations formed by disjunctions of basic relations, *eg.* $I_1\{m\ o\}I_2$ denotes that either I_1 meets I_2 or I_1 overlaps I_2. This gives rise to a total number of $2^{13} = 8192$ basic and composite relations.

A somewhat less common algebra is the *point-interval algebra* (and its converse, the interval-point algebra) [Vilain, 1982], which allows only relations between a time point and an interval. The point-interval algebra has five basic relations, *before*, *starts*, *during*, *ends* and *after*, as shown in Figure 6.3. The point-interval algebra thus has a total of $2^5 = 32$ relations, basic and composite. Note that it is not possible to directly relate two time points or two intervals to each other, which makes its usage somewhat limited. However, it is useful in at

[3]In fact, Allen was not first to consider interval relations. Bruce [1972] defines many, though not all, of Allen's interval relations. In fact, Bruce also allows intervals which are points, thus not being a proper subclass of the interval algebra.

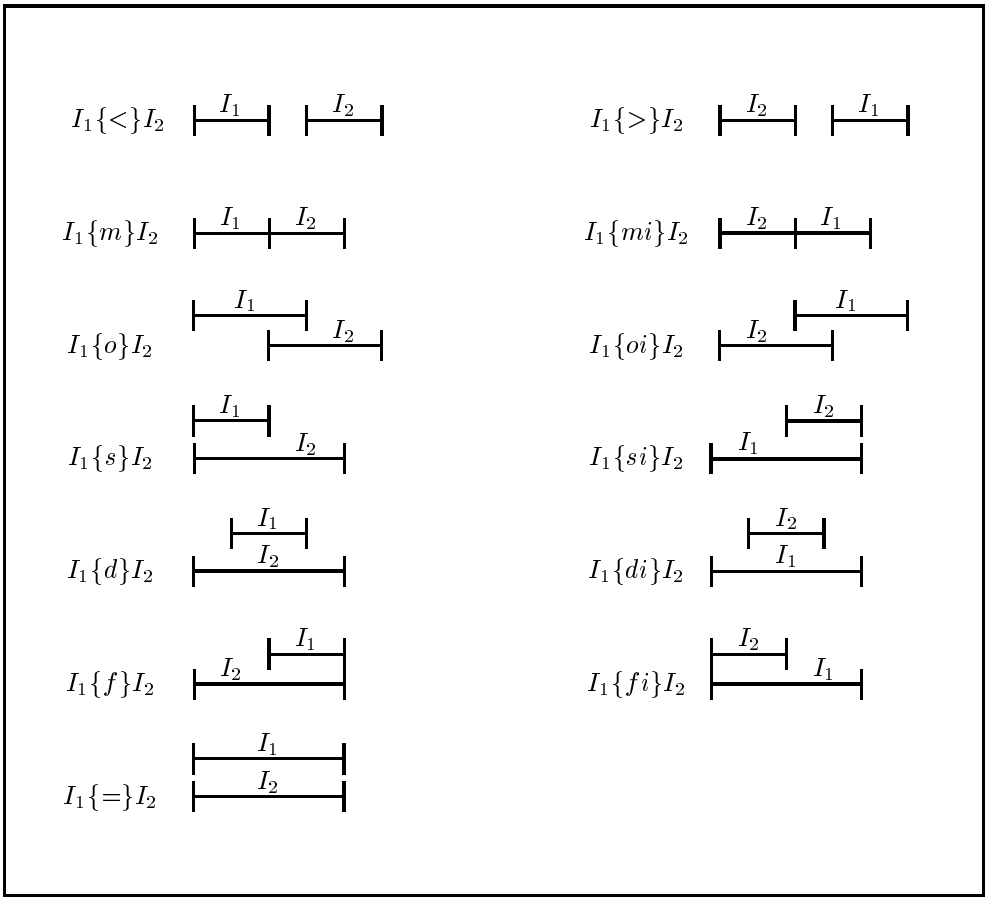

Figure 6.2: The 13 basic relations in the interval algebra.

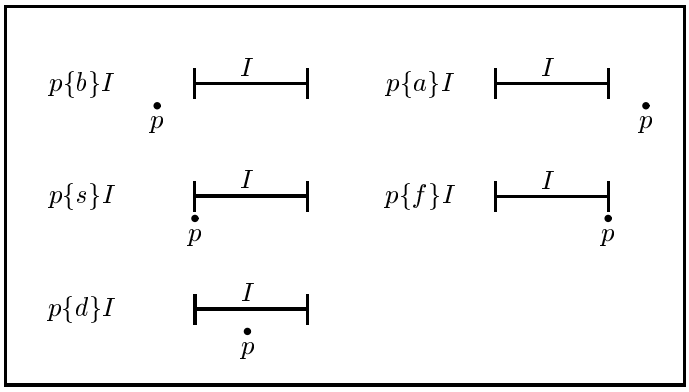

Figure 6.3: The five basic relation in the point-interval algebra.

least two contexts. It is used in Meiri's QA algebra [Meiri, 1991] as a 'glue' for integrating reasoning about time points and intervals. Furthermore, it is sufficient to express information about partially ordered plans.[4] A consequence of this asymmetry is further that not all of the normal algebraic operations can be defined in the usual way. In fact, Vilain [1982] does not discuss this problem, but an alternative to transitivity, *3-composition*, has recently been introduced in the literature [Jonsson *et al.*, 1996].

Quantitative Algebras

A number of quantitative temporal algebras have been proposed in the literature, most or all of them based on time points. The ancestor of most of these algebras is the *time map manager* (TMM) [Dean and McDermott, 1987], which maintains a network of time points and metric relations between these. A relation consists of upper and lower bounds on the temporal distance between two time points. This is what Dechter *et. al.* [1991] refers to as a *binary constraint*, which can be written as $c \leq t_1 - t_2 \leq d$, where t_1 and t_2 are time points and c and d are constants. One may also wish to locate a single time point within an interval on the time line, which can be done via a *unary constraint* of the form $c \leq t \leq d$. Note, however, that by introducing a reference time point t_{ref} defined to occur at time 0 a unary constraint $c \leq t \leq d$ can always be replaced by the binary constraint $c \leq t - t_{ref} \leq d$. Further note that the expressions $t = c$ and $t_1 - t_2 = c$ can be expressed as the unary and binary constraints $c \leq t \leq c$ and $c \leq t_1 - t_2 \leq c$ respectively. Most quantitative algebras are based on sets of unary and/or binary constraints, sometimes augmented also with inequality formulae of the forms $t \neq c$ and $t_1 - t_2 \neq c$. Some algebras use only one-sided constraints, that is, constraints of the forms $t \leq d$ and $t_1 - t_2 \leq d$. This is no restriction, however, since a two-sided constraint corresponds to two one-sided constraints. Furthermore, most of the quantitative algebras are really mixed qualitative and quantitative algebras in sense that they can express some, or all, qualitative relations over time points, *ie.* they subsume the point algebra. Some of those not subsuming the whole point algebra are augmented with the full set of qualitative relations in order to do so.

There are also more expressive algebras, allowing more complex relations and/or more complex relationships between relational expressions. For instance, the *temporal constraint satisfaction problem* (TCSP) [Dechter *et al.*, 1991] allows a set of formulae which are either disjunctions of unary constraints or disjunctions of binary constraints, restricted to one time point/pair of time points per constraint. That is, each formula is either of the form

$$(c_1 \leq t_i \leq d_1) \vee \ldots \vee (c_n \leq t_i \leq d_n)$$

[4] If treating the actions as time points and the temporal intervals between them as intervals, both the partial order between the action and exclusion of actions from protection intervals can be modelled.

or

$$(c_1 \leq t_i - t_j \leq d_1) \vee \ldots \vee (c_n \leq t_i - t_j \leq d_n).$$

A set of TCSPs obviously correspond to a formula in conjunctive normal form with binary temporal constraints as atoms.

A more general approach which subsumes TCSP and extends it in several ways, for instance by allowing inequalities and more-than-binary constraints, is is the language of *disjunctive linear relations* (DLRs) [Jonsson and Bäckström, 1996b]. A DLR is a disjunction of linear relations, the latter being expressions on the form $\alpha\ r\ \beta$, where α and β are linear expressions (*ie.* polynomials of degree one) with rational coefficients over the set of time points and $r \in \{<, \leq, =, \geq, >, \neq\}$. For instance, the formula

$$(3t_1 + 5.42t_4 - 0.7t_5 < 12) \ \vee \ (t_2 + 3t_3 + 0.55t_4 \geq t_7) \ \vee \ (t_1 + 2t_3 \neq t_4)$$

is a DLR. That is, a set of DLRs is a formula in conjunctive normal form over linear relations, so it constitutes a full propositional logic over the set of linear relations and is powerful enough to subsume all other algebras, qualitative as well as quantitative, discussed so far. Note, for instance, that DLRs are sufficient for encoding all relations in the interval algebra, which is not possible using only conjunctions of linear relations or unary/binary constraints. Hence, the DLR formalism is strictly more expressive than TCSP formalism.[5]

Kautz and Ladkin [1991] define a language for mixed qualitative and metric temporal reasoning which contains both intervals and time points. All interval relations are allowed between intervals, while time points and differences between two time points can be related to constants using either $<$ or $\leq$. Furthermore, all interval end-points are by definition in the set of time points, thus forming a connection between interval formulae and time-point formulae. The actual reasoning system uses separate reasoners for time-point relations and interval relations, and uses translation algorithms to cross-post constraints between these two reasoners. A similar approach is taken by Meiri [1991] who extends his qualitative algebra QA with metric reasoning capabilities.

Computational Issues

While the choice of ontology and a particular algebra certainly affects how easy it is to model a particular problem, and even whether it is possible at all, these choices also have computational implications. Different temporal algebras may have strikingly different computational properties, so it is interesting to study various restricted algebras.

[5] Actually, from a computational point of view this is not quite true, as will be seen below, but there seems to be no straightforward, obvious way of transforming a set of DLRs into a set of TCSPs.

Reasoning Problems

Before discussing the complexity of reasoning in an algebra, we have to define exactly what reasoning problem we are analysing. Several interesting reasoning problems can be defined, but the most common and basic one is the *satisfiability problem*. Given a set of relational expressions, interpreted as the conjunction of these expressions, over some temporal algebra the satisfiability problem asks whether this set of expressions is satisfiable—that is, whether there exists an *interpretation* (an assignment of exact times to all time points/interval end points) such that all expressions are simultaneously true. Such an interpretation is referred to as a *model*.

Another common reasoning problem is the *minimal labelling problem*: given a set of temporal entities and relations between these, decide for two particular entities what is the least constrained relation that may hold between these s.t. all other constraints are still satisfied.

Reasoning with Qualitative Algebras

It has been known for a number of years that the satisfiability problem for the point algebra can be solved in polynomial time (*ie.* it is computationally *tractable*) [van Beek and Cohen, 1990].[6]

The satisfiability problem for the interval algebra, on the other hand, is known to be NP-complete [Kautz and Ladkin, 1991]. Intuitively, this is hardly surprising since we cannot rewrite an arbitrary set of interval relations as a pure conjunction of point relations. For instance, the relation $I_1\{<>\}I_2$ must be encoded as the disjunction $(I_1{}^+ > I_2{}^-) \vee (I_2{}^+ < I_1{}^-)$.[7] Despite its apparently lower expressivity, also the point-interval algebra is known to be NP-complete with respect to satisfiability [Meiri, 1991].

The NP-completeness result for the interval algebra has provoked research into identifying subalgebras where the satisfiability problem is tractable. The most well-known such subalgebras in the literature are the *pointisable algebra* [van Beek and Cohen, 1990], the fragment of the interval algebra that can be reencoded in the point algebra, and the *ORD-Horn algebra* [Nebel and Bürckert, 1995], the fragment of the interval algebra which can be reencoded as Horn clauses of the

[6] Indeed, this problem can be solved in linear time in the number of constraints) [Gerevini *et al.*, 1993].

[7] Actually, the relation $\{<>\}$ alone is not a sufficient cause for NP-completeness. More interestingly, the seemingly harmless relation meets alone is sufficient to express all 13 basic relations, and together with either $<$ or $>$ it causes NP-completeness—the relation $\{< \ m\}$ generates the whole interval algebra. It is further interesting to note that the relation $\{<>\}$ in the point algebra does not cause any computational problem since it excludes only a point, not an interval, on the time line. This can be understood as follows. First note that the relations $\leq$ and $\neq$ are sufficient to replace all other relations of the point algebra. Now, any cycle of $\leq$ relations is unsatisfiable if any two time points in the cycle are related by a $\neq$ constraint and otherwise is satisfiable and can be collapsed to a single time point.

relations $\leq$, $=$ and $\neq$ over time points.[8] The ORD-Horn algebra is further interesting since it is proven to be the unique maximal tractable subalgebra allowing all 13 basic relations (but obviously not all disjunctions of these). The ORD-Horn algebra was claimed better than the pointisable algebra, since it contains 868 elements (*ie.* more than 10% of the full algebra) in contrast to the 188 relations covered by the pointisable algebra. However, recent research by Drakengren and Jonsson [1996a, 1996b] has identified 17 new maximal tractable subalgebras, all of them containing considerably more relations than the ORD-Horn algebra. As is obvious from the maximality result for the ORD-Horn algebra, none of these new algebras properly includes the ORD-Horn algebra or contains all the basic relations. Furthermore, these new results also question the value of using the size of a subalgebra as a measure for its goodness; the largest of the new algebras contains half of the full interval algebra, but cannot express anything of interest. It is not known how many tractable subalgebras that exist, and this is a non-trivial question to answer—the interval algebra has $2^{8192} \approx 10^{244}$ subclasses.

On the other hand, a complete classification of all subalgebras of the point-interval algebra into tractable and NP-complete respectively has recently been reported [Jonsson *et al.*, 1996]—nine out of the 2^{32} subclasses being maximal tractable subalgebras.

Furthermore, Golumbic and Shamir [1993] have studied algebras formed by collapsing sets of basic relations into macro relations. For instance, one such algebra has the relations $\{<, \cap, >\}$ where $\cap$ denotes the set of all the other 11 basic interval relations. They report a number of tractability and intractability results, but it is worth noting they do not always allow two intervals to be unrelated, making comparisons with other work somewhat difficult.

Quantitative Algebras

Satisfiability for a set of DLRs is NP-complete [Jonsson and Bäckström, 1996b], which is obvious since the interval algebra can be encoded as a set of DLRs. On the other hand, there is no straightforward encoding of interval-algebra satisfiability into TCSP—yet, TCSP is also NP-complete. Hence, it is obviously interesting to study tractable subalgebras also in the case of quantitative temporal reasoning.

Various tractable algebras based on unary/binary constraints, often augmented with inequalities and/or qualitative relations have been reported in the literature, including the following:

- Dechter *et. al.* [1991] consider sets of *simple temporal constraints* (STP), a restricted version of TCSP allowing only formulae of the form $c \leq (x - y) \leq d$, where x and y are time points and c and d constants. Simple temporal

[8] The term Horn clause is here generalized to mean a disjunction of relations where at most one relation is $\leq$ or $=$.

constraints are equivalent to *simple metric constraints* [Kautz and Ladkin, 1991] and the constraints use in Barber's [1993] system.

- Koubarakis [1992] considers sets of formulae on either of the three forms (1) $(x - y)\, r\, c$, (2) $x\, r\, c$ or (3) a disjunction of formulae of the form $(x - y) \neq c$ or $x \neq c$, where $r \in \{\leq, \geq, \neq\}$.

- Meiri [1991] considers sets of *CPA/single interval* formulae, that is, formulae on either of the two forms (1) $c\, r_1\, (x - y)\, r_2\, d$; or (2) $x\, r\, y$ where $r \in \{<, \leq, =, \neq, \geq, >\}$ and $r_1, r_2 \in \{<, \leq\}$.

- The TimeGraph II system [Gerevini *et al.*, 1993] allows formulae on either of the three forms (1) $c \leq x \leq d$, (2) $c \leq x - y \leq d$ or (3) $x\, r\, y$ where $r \in \{<, \leq, =, \neq, \geq, >\}$.

A recent exception is the Horn-DLR formalism by Jonsson and Bäckström [1996a, 1996b], which is the DLR formalism restricted to Horn DLRs, that is, DLRs where at most one relational expression contains the $\leq$ relation. As an example, the two formulae

$$(t_1 \neq t_2) \ \vee \ (t_2 \neq 7) \ \vee \ (t_1 = t_3 - 4)$$
$$(3t_1 \neq t_2 - 4) \ \vee \ (t_1 + 2t_2 - 1 \neq 1.4t_3) \ \vee \ (t_1 + 4t_3 \leq t_2 + 7)$$

are both Horn-DLR formulae, while the formula

$$(t_1 \neq t_2) \ \vee \ (t_1 \leq 4) \ \vee \ (t_2 \leq t_4)$$

is a DLR, but not a Horn-DLR.

Satisfiability for a set of Horn DLRs can be solved in polynomial time using an algorithm based on linear-programming. The Horn-DLR formalism subsumes all of the tractable quantitative algebras listed above, thus serving as a unifying formalism for these. Furthermore, the Horn DLR formalism also subsumes some of the tractable interval subalgebras, including the ORD-Horn algebra. Hence, the qualitative fragment of Horn DLR inherits the maximality result for the ORD-Horn algebra, while at the same time adding a capability of reasoning about metric time, thus constituting a mixed qualitative/quantitative tractable algebra. (The proof of tractability for the Horn DLR formalism will be provided as an example in the following section.) There is also recent work on integrating other tractable interval subalgebras with quantitative temporal reasoning using the Horn DLR formalism [Drakengren and Jonsson, 1996b]. The algorithm for the Horn DLR algebra is, although polynomial, less efficient than the specialized algorithms for many of the less expressive tractable algebras listed above. In contrast to these specialized algorithms, however, the Horn-DLR algorithm works for all these algebras.

6.2 Some Examples in Detail

The previous section mostly discusses ontological and computational aspects of temporal algebras in a somewhat coarse and survey-like style. Hence, this chapter provides two examples of proofs of claims from the previous section in order to allow the interested reader to get a flavour of the character of such proofs and the computational issues involved. Since the major computational question to ask about satisfiability in a temporal algebra (as about any other problem) is whether it can be proven computationally tractable or intractable[9] one proof of each type is provided here. The first of these proves that satisfiability is tractable for the Horn-DLR formalism, which is done by means of providing an algorithm and proving that it is correct and polynomial time. The other proof, which is of the opposite type, shows that the subalgebra $\{\{s\ f\}, \{b\ d\ a\}\}$ of the point-interval algebra is NP-complete.

Tractability for the Horn-DLR Algebra

The following material oringally appears in Jonsson and Bäckström [1996a].

Disjunctive Linear Relations

We begin by defining som different types of linear relations.

Definition 6.2.1 *Let $X = \{x_1, \ldots, x_n\}$ be a set of real-valued variables. Let α, β be linear polynomials (ie. polynomials of degree one) over X. A* linear disequation *over X is an expression of the form $\alpha \neq \beta$. A* linear equality *over X is an expression of the form $\alpha = \beta$. A* linear relation *over X is an expression of the form $\alpha r \beta$ where $r \in \{<, \leq, =, \neq, \geq, >\}$. A* convex linear relation *over X is an expression of the form $\alpha r_c \beta$ where $r_c \in \{<, \leq, =, \geq, >\}$. A* disequational linear relation *over X is an expression of the form $\alpha \neq \beta$. A* disjunctive linear relation *(DLR) is a disjunction of one or more linear relations.*

Example 6.2.1 *A typical DLR over $\{x_1, x_2, x_3\}$ is $(1.2x_1 + x_2 \leq x_3 + 5) \vee (12x_3 \neq 7.5x_2) \vee (x_2 = 5)$.*

In the following, we assume all sets of DLRs to be finite. The definition of satisfiability for DLRs is straightforward.

Definition 6.2.2 *Let $X = \{x_1, \ldots, x_n\}$ be a set of real-valued variables and let $R = \{R_1, \ldots, R_k\}$ be a set of DLRs over X. We say that R is* satisfiable *iff there exists an assignment of real values to the variables in X that makes at least one member of each R_i, $1 \leq i \leq k$, true.*

[9] It may, of course, be quite difficult to provide such proofs, and some problems have so far defied attempts to prove either case. Further, the term intractable must usually be taken liberally, meaning intractable under the assumption that $P \neq NP$.

It is important to note that we only consider assignments of *real* values, thus assuming that time is linear, dense and unbounded. (We will see that it is sufficient to consider assignments of rational values further on.) We continue by classifying different types of DLRs.

Definition 6.2.3 *Let γ be a DLR. $\mathcal{C}(\gamma)$ denotes the convex relations in γ and $\mathcal{NC}(\gamma)$ the disequational relations in γ. We say that γ is* convex *iff $|\mathcal{NC}(\gamma)| = 0$ and that γ is* disequational *iff $|\mathcal{C}(\gamma)| = 0$. If γ is convex or disequational we say that γ is* homogeneous *and otherwise* heterogeneous. *Furthermore, if $|\mathcal{C}(\gamma)| \leq 1$ then γ is* Horn. *We extend these definitions to sets of relations in the obvious way. For example, if Γ is a set of DLRs and all $\gamma \in \Gamma$ are Horn, then Γ is Horn.*

This classification provides the basis for the forthcoming proofs. One detail to note is that if a Horn DLR is convex then it is a unit clause, *ie.* a disjunction with only one member.

For Horn DLRs, we restrict ourselves only to use $\leq$ and $\neq$ in the relations. This is no loss of generality since we can express all the other relations in terms of these two. For example, a DLR of the form $x < y \vee D$ can be replaced by the disjunctions $\{x \leq y \vee D, x \neq y \vee D\}$. Observe that the resulting set of disjunctions can contain at most twice as many disjunctions as the original one. Hence, this is a polynomial time transformation. (Note, however, that this does not hold for general DLRs.)

Definition 6.2.4 *Let A be a satisfiable set of DLRs and let γ be a DLR. We say that γ* blocks *A iff for every $d \in \mathcal{NC}(\gamma)$, $A \cup \{d\}$ is not satisfiable.*

Observe that if $A \cup \{\gamma\}$ is satisfiable and γ blocks A then there must exist a relation $\delta \in \mathcal{C}(\gamma)$ such that $A \cup \{\delta\}$ is satisfiable. This observation will be of great importance in forthcoming sections.

Linear Programming

The method for deciding satisfiability of Horn DLRs is based on linear-programming techniques so some of the basic facts about linear programming are repeated here for the readers convenience.

Definition 6.2.5 *Let A be an arbitrary $m \times n$ matrix of rationals in finite precision and let $x = (x_1, \ldots, x_n)$ be an n-vector of variables over the real numbers. Then an instance of the* linear programming *(LP) problem is defined by: $\{\min c^T x$ subject to $Ax \leq b\}$ where b is an m-vector of rationals and c an n-vector of rationals. The computational problem is as follows:*

1. *Find an assignment to the variables $x_1, \ldots, x_n$ such that the condition $Ax \leq b$ holds and $c^T x$ is minimial subject to these conditions, or*

1 **algorithm** SAT(Γ)
2 $A \leftarrow \bigcup\{\mathcal{C}(\gamma) | \gamma \in \Gamma \text{ is convex}\}$
3 **if** A not satisfiable **then reject**
4 **if** $\exists\gamma \in \Gamma$ that blocks A and is disequational **then reject**
5 **if** $\exists\gamma \in \Gamma$ that blocks A and is heterogeneous **then** SAT($(\Gamma - \{\gamma\}) \cup \mathcal{C}(\gamma)$)
6 **accept**

Figure 6.4: Algorithm for deciding satisfiability of Horn DLRs.

2. *Report that there is no such assignment, or*

3. *Report that there is no lower bound for $c^T x$ under the conditions.*

Analogously, we can define an LP problem where the objective is to maximize $c^{\mathrm{T}} x$ under the condition $Ax \leq b$. We have the following important theorem.

Theorem 6.2.6 *[Khachiyan, 1979] The linear programming problem is solvable in polynomial time.*

Although polynomial, Khachiyans algorithm was not very efficient, and in practice it was often outperformed by the non-polynomial Simplex method. More recent work, starting with the algorithm by Karmarkar [1984], has changed this however and the polynomial methods are now generally considered the best also in practice. In the following, we assume all coeffecients to be rationals represented in finite precision, which is no restriction in practice since computers (almost without exception) use finite precision arithmetics.

Satisfiability of Horn DLRs

The problem of deciding satisfiability for a set of Horn DLRs is denoted HornDLRSat and we will show below that this problem can be solved in polynomial time using the algorithm SAT (Figure 6.4).

We begin by exhibiting a simple method for deciding whether a set of convex linear relations augmented with one disequation is satisfiable or not.

Note that for this methods, as well as for the final algorithm, the purpose has only been to prove tractability, and not to attempt finding the best possible upper bound. That is, simplicity is stressed rather than tuning of efficiency.

Lemma 6.2.7 *Let A be an arbitrary $m \times n$ matrix, b be an m-vector and $x = (x_1, \ldots, x_n)$ be an n-vector of variables over the real numbers. Let α, β be linear polynomial over $x_1, \ldots, x_n$. Deciding whether the system $S = \{Ax \leq b, \alpha \neq \beta\}$ is satisfiable or not is polynomial.*

Proof: Let $\alpha' = \alpha - c$ and $\beta' = \beta - d$ where c and d are the constant terms in α and β, respectively. Consider the following instances of LP:

LP1= $\{\min\ \alpha' - \beta' \text{ subject to } Ax \leq b\}$

LP2= $\{\max\ \alpha' - \beta' \text{ subject to } Ax \leq b\}$

If LP1 and LP2 have no solutions then S is not satisfiable. If both LP1 and LP2 yield the same optimal value $d - c$ then S is not satisfiable since every solution to LP1 and LP2 forces α to equal β. Otherwise S is obviously satisfiable. Since we can solve the LP problem in polynomial time by Theorem 6.2.6, the lemma follows. □

Before proceeding, we recapitulate some standard mathematical concepts.

Definition 6.2.8 *Given two points $x, y \in R^n$, a* convex combination *of them is any point of the form $z = \lambda x + (1 - \lambda)y$ where $0 \leq \lambda \leq 1$. A set $S \subseteq R^n$ is* convex *iff it contains all convex combinations of all pairs of points $x, y \in S$.*

Definition 6.2.9 *A* hyperplane *H in R^n is a non-empty set defined as $\{x \in R^n \mid a_1x_1 + \ldots + a_nx_n = b\}$ for some $a_1, \ldots, a_n, b \in R$.*

Definition 6.2.10 *Let A be an arbitrary $m \times n$ matrix and b be an m-vector. The* polyhedron *defined by A and b is the set $\{x \in R^n | Ax \leq b\}$.*

The connection between polyhedrons and convex sets is expressed in the following well-known fact.

Fact 6.2.11 *Every non-empty polyhedron is convex.*

Consequently, the convex relations in a set of Horn DLRs defines a convex set in R^n. Furthermore, we can identify the disequations with hyperplanes in R^n. These observations motivate the next lemma.

Lemma 6.2.12 *Let $S \subseteq R^n$ be a convex set and let $H_1, \ldots, H_k \subseteq R^n$ be distinct hyperplanes. If $S \subseteq \bigcup_{i=1}^{k} H_i$ then there exists a j, $1 \leq j \leq k$ such that $S \subseteq H_j$.*

Proof: If it is possible to drop one or more hyperplanes from H and still have a union containing S then do so. Let $H' = \{H'_1, \ldots, H'_m\}$, $m \leq k$, be the resulting minimal set of hyperplanes. Every $H'_i \in H'$ contains some point x_i of S not in any other $H'_j \in H'$. We want to prove that there is only one hyperplane in H'.

If this is not the case, consider the line segment L adjoining x_1 and x_2. (The choice of x_1 and x_2 is not important. Every choice of x_i and x_j, $1 \leq i, j \leq m$ and $i \neq j$, would do equally well.) By convexity, $L \subseteq S$. Each $H'_i \in H'$ either contains L or meets it in at most one point. But no $H'_i \in H'$ can contain L, since then it would contain both x_1 and x_2. Thus each H'_i has at most one point in common with L, and the rest of L would not be a subset of $\bigcup_{i=1}^{m} H'_i$ which contradicts that $L \subseteq S \subseteq \bigcup_{i=1}^{m} H'_i$. □

We can now tie together the results and end up with a sufficient condition for satisfiability of Horn DLRs.

Lemma 6.2.13 *Let Γ be a set of arbitrary Horn DLRs. Let $C \subseteq \Gamma$ be the set of convex DLRs in Γ and let $D = \{D_1, \ldots, D_k\} \subseteq \Gamma$ be the set of DLRs that are not convex. Under the condition that C is satisfiable, Γ is satisfiable if D_i does not block C for any $1 \leq i \leq k$.*

Proof: Pick one disequation d_i out of every D_i such that $\{C, d_i\}$ is satisfiable. This is possible since no D_i blocks C. We show that $\Gamma' = \{C, d_1, \ldots, d_k\}$ is satisfiable and, hence, Γ is satisfiable. Assume that $d_i = (\alpha_i \neq \beta_i)$. Define the hyperplanes $H_1, \ldots, H_k$ such that $H_i = \{x \in R^n \mid \alpha_i(x) = \beta_i(x)\}$. Since every $\{C, d_i\}$ is satisfiable, the polyhedron P defined by C (which is non-empty and hence convex by Fact 6.2.11) is not a subset of any H_i. Suppose Γ' is not satisfiable. Then $P - \bigcup_{i=1}^{k} H_i = \varnothing$ which is equivalent with $P \subseteq \bigcup_{i=1}^{k} H_i$. By Lemma 6.2.12, there exists a H_j, $1 \leq j \leq k$ such that $S \subseteq H_j$. Clearly, this contradicts our initial assumptions. □

It is important to note that the previous lemma does not give a necessary condition for satisfiability of Horn DLRs, so it remains to prove that the algorithm in Figure 6.4 correctly solves HornDLRSat in polynomial time. To show this, we need an auxiliary lemma which is a formal version of an earlier observation.

Lemma 6.2.14 *Let Γ be a set of Horn DLRs and let $C \subseteq \Gamma$ be the set of convex DLRs in Γ. If there exists a heterogeneous DLR $\gamma \in \Gamma$ such that γ blocks C, then Γ is satisfiable iff $(\Gamma - \{\gamma\}) \cup \mathcal{C}(\gamma)$ is satisfiable.*

Proof: *if:* Trivial.
only-if: If Γ is satisfiable, then γ has to be satisfiable. Since γ blocks C, $\mathcal{C}(\gamma)$ must be satisfied in any solution of Γ. □

We can now prove the soundness and completeness of SAT.

Lemma 6.2.15 *Let Γ be a set of Horn DLRs. If SAT(Γ) accepts then Γ is satisfiable.*

Proof: Induction over n, the number of heterogeneous DLRs in Γ.
Basis step: If $n = 0$ and SAT(Γ) accepts then the formulae in A are satisfiable and there does not exist any $\gamma \in \Gamma$ that blocks A. Consequently, Γ is satisfiable by Lemma 6.2.13.
Induction hypothesis: Assume the claim holds for $n = k$, $k \geq 0$.
Induction step: Γ contains $k + 1$ heterogeneous DLRs. If SAT accepts in line 5 then $(\Gamma - \{\gamma\}) \cup \mathcal{C}(\gamma)$, which contains k heterogeneous DLRs, is satisfiable by the induction hypothesis. By Lemma 6.2.14, this is equivalent with Γ being satisfiable. If SAT accepts in line 6 then there does not exist any disequational or heterogeneous $\gamma \in \Gamma$ which blocks A. By Lemma 6.2.13, this means that Γ is satisfiable. □

Before proving the completeness of SAT we need the following lemma.

Lemma 6.2.16 *Let Γ be a set of Horn DLRs. Let $C \subseteq \Gamma$ be the set of convex DLRs in Γ. If there exists a disequational DLR $\gamma \in \Gamma$ that blocks C then Γ is not satisfiable.*

Proof: In any solution to Γ, the relations in $C \cup \{\gamma\}$ must be satisfied. Since γ is disequational and blocks C this is not possible and the lemma follows. □

Lemma 6.2.17 *Let Γ be a set of Horn DLRs. If SAT(Γ) rejects then Γ is not satisfiable.*

Proof: Induction over n, the number of heterogeneous DLRs in Γ.
Basis step: If $n = 0$ then SAT can reject in lines 3 and 4. If SAT rejects in line 3 then, trivially, Γ is not satisfiable. If SAT rejects in line 4 then there exists a disequational $\gamma \in \Gamma$ that blocks A. Hence, Γ is not satisfiable by Lemma 6.2.16.
Induction hypothesis: Assume the claim holds for $n = k$, $k \geq 0$.
Induction step: Γ contains $k + 1$ heterogeneous DLRs. If SAT rejects in line 3 then Γ is not satisfiable. If SAT rejects in line 4 then Γ is not satisfiable by Lemma 6.2.16. If SAT rejects in line 5 then $(\Gamma - \{\gamma\}) \cup \mathcal{C}(\gamma)$, which contains k heterogeneous DLRs, is not satisfiable by the induction hypothesis. By Lemma 6.2.14, this is equivalent with Γ not being satisfiable. □

Finally, we can show that SAT is a polynomial-time algorithm and, thus, show that HornDLRSat is polynomial.

Theorem 6.2.18 HornDLRSat *is polynomial.*

Proof: By Lemmata 6.2.15 and 6.2.17, it is sufficient to show that SAT is polynomial. The number of recursive calls is bounded by the number of heterogeneous DLRs in the given input. By Lemma 6.2.7, we can in polynomial time decide whether a linear inequality system with one disequation is satisfiable. Since we need only check a polynomial number of such systems in each recursion, the theorem follows. □

Comparison

Below we show that Horn DLRs subsumes several other tractable methods for temporal constraint reasoning. Let x, y be real-valued variables, c, d constants and $\mathcal{A}$ Allen's algebra [Allen, 1983] in the definitions below.

Definition 6.2.19 *[Nebel and Bürckert, 1995] An* ORD clause *is a disjunction of relations of the form xry where $r \in \{\leq, =, \neq\}$. The* ORD-Horn *subclass $\mathcal{H}$ is the relations in $\mathcal{A}$ that can be written as ORD clauses containing only disjunctions with at most one relation of the form $x = y$ or $x \leq y$ and an arbitrary number of relations of the form $x \neq y$.*

Note that the ORD-Horn class subsumes the pointisable endpoint algebra [van Beek and Cohen, 19

Definition 6.2.20 *[Koubarakis, 1992] Let* $r \in \{\leq, \geq, \neq\}$. *A* Koubarakis formula *is a formula on one of the following forms (1)* $(x - y)rc$*, (2)* xrc *or (3) a disjunction of formulae of the form* $(x - y) \neq c$ *or* $x \neq c$.

Definition 6.2.21 *[Dechter* et al.*, 1991] A* simple temporal constraint *is a formula on the form* $c \leq (x - y) \leq d$.

Simple temporal constraints are equivalent with the *simple metric constraints* [Kautz and Ladkin, 1991].

Definition 6.2.22 *[Meiri, 1991] A* CPA/single interval *formula is a formula on one of the following forms: (1)* $c\, r_1\, (x - y)\, r_2\, d$*; or (2)* $x\, r\, y$ *where* $r \in \{<, \leq, =, \neq, \geq, >\}$ *and* $r_1, r_2 \in \{<, \leq\}$.

Definition 6.2.23 *[Gerevini* et al.*, 1993] A* TG-II *formula is a formula on one of the following forms: (1)* $c \leq x \leq d$*, (2)* $c \leq x - y \leq d$ *or (3)* $x\, r\, y$ *where* $r \in \{<, \leq, =, \neq, \geq, >\}$.

We can now state the main theorem of this section.

Theorem 6.2.24 *The formalisms defined in Definitions 6.2.19 to 6.2.23 can trivially be expressed as Horn DLRs.*

Note that Meiri [1991] considers two further tractable classes that cannot (in any obvious way) be transformed into Horn DLRs. The finding that the ORD-Horn algebra can be expressed as Horn DLRs is especially important in the light of the following theorem.

Theorem 6.2.25 *[Nebel and Bürckert, 1995] Let* $\mathcal{S}$ *be any subclass of* $\mathcal{A}$ *that contains all basic relations. Then either*

1. $\mathcal{S} \subseteq \mathcal{H}$ *and the satisfiability problem for* $\mathcal{S}$ *is polynomial, or*
2. *Satisfiability for* $\mathcal{S}$ *is NP-complete.*

By the previous theorem, we cannot expect to find tractable classes that are able to handle all basic relations in $\mathcal{A}$ and, at the same time, are able to handle any single relation that cannot be expressed as a Horn DLR. In other words, the qualitative fragment of HornDLRSat inherits the maximality of the ORD-Horn algebra.

As a concluding discussion to this comparison it seems in place to mention that several researchers in the field of temporal constraint reasoning have expressed a feeling that their proposed methods should be extended so they can express

relations between more than two time points. As a first example, Dehter *et. al.* [1991] write "The natural extension of this work is to explore TCSPs with higher-order expressions (e.g. "John drives to work at least 30 minutes more than Fred does"; $X_2 - X_1 + 30 \leq X_4 - X_3$)..." Even though they do not define the exact meaning of "higher-order expressions" we can notice that their example is a simple Horn DLR. Something similar can be found in Koubarakis [1992] who wants to express "the duration of interval I exceeds the duration of interval J". Once again, this can easily be expressed as a Horn DLR. These claims seem to indicate that the use of Horn DLRs is a significant contribution to temporal reasoning.

NP-completeness of the $\{\{s\ f\}, \{b\ d\ a\}\}$ Algebra

While the previous subsection presented an example where we could prove tractability for an algebra, this subsection provides an example of an algebra that is rather proven computationally difficult. More precisely, we reproduce a proof from Jonsson *et.al.* [1996] that the subalgebra of the point-interval algebra that allows only the two relations $\{b\ d\ a\}$ and $\{s\ f\}$ is NP-complete.[10]

Theorem 6.2.26 *Satisfiability is NP-complete for the point-interval subalgebra* $\{\{b\ d\ a\}, \{s\ f\}\}$.

Proof: Proof by reduction from GRAPH 3-COLOURABILITY, which is NP-complete. Let $G = \langle V, E\rangle$ be an undirected graph. Construct a corresponding set of temporal constraints as follows.

In the proof We will make repeated use of the concept of a *separator*, a construction which forces two points to have distinct values in all models. Given two points p, q we construct a separator by introducing a new interval I and adding the relations $p\{s\ f\}I$ and $q\{b\ d\ a\}I$. Clearly, all models $\mathcal{M}$ must satisfy $\mathcal{M}(p) \neq \mathcal{M}(q)$.

We now construct set of temporal constraints stepwise. First, we construct a *paint-box* by introducing two points p_1 and p_2, two intervals I_1 and I_2 plus the relations

$$p_1\{s\ f\}I_1,\ p_1\{s\ f\}I_2,\ p_2\{s\ f\}I_1,\ p_2\{b\ d\ a\}I_2$$

over these. Note that the interval I_2 acts as a separator for p_1 and p_2, which are thus forced to take on different values. Further, the intervals I_1 and I_2 must have some common end-point, coinciding with p_1. We use the constant r to denote this value. Hence, the remaining end-point of I_1 must coincide with p_2 and the

[10]The proof that satisfiability for a set of DLRs is NP-complete would, of course, have provided a natural continuation of the previous example. However, this proof is one of the rare examples where it is trivial to prove NP-completeness for a temporal algebra, and so is better left as an exercise for the reader (or see Jonsson and Bäckström [Jonsson and Bäckström, 1996b]) in order to leave space for the more intricate and interesting proof presented here.

remaining end-point of I_2 must be distinct from both p_1 and p_2. We denote the values of these two remaining end-points g and b respectively. We can think of the values r, g and b as colours, constituting our palette. Of course, the actual denotations of these three values differ between models, but the important thing is only that they denote three distinct values in each and every model.

Now, for each vertex $v_i \in V$, we construct a *selector* consisting of three points q_i^0, q_i^1 and q_i^2 plus two intervals J_i^0 and $J_i^{1,2}$, connected as follows. First introduce a separator for q_i^1 and q_i^2, using interval $J_i^{1,2}$, *ie.* introduce the relations

$$q_i^1\{s\ f\}J_i^{1,2},\ q_i^2\{b\ d\ a\}J_i^{1,2}.$$

Then connect the points to the remaining interval by adding the relations

$$q_i^0\{s\ f\}J_i^0,\ q_i^1\{s\ f\}J_i^0,\ q_i^2\{s\ f\}J_i^0.$$

Finally, connect this whole gadget to the paint-box by adding the relations

$$q_i^1\{s\ f\}I_1,\ q_i^2\{s\ f\}I_2.$$

The selector works as follows. The endpoints of I_1 correspond to the colours r and g, so q_i^1 is forced to have either of these values. Similarly, q_i^2 must have either of the values r and b. Now, q_i^1 and q_i^2 are separated, so together they select a subpalette of two colours, assigning one colour each to the end-points of J_i^0. Finally, q_i^0 selects one of these two colours. So far, there are no further constraints, so q_i^0 may be freely assigned any of the three colours from our palette.

Finally, for each edge $\{v_i, v_j\} \in E$ we introduce a separator, consisting of the new interval $K_{i,j}$ and the two relations

$$q_i^0\{s\ f\}K_{i,j},\ q_j^0\{b\ d\ a\}K_{i,j},$$

preventing q_i^0 and q_j^0 to have the same value whenever there is an edge between the vertices v_i and v_j.

It is obvious that G is 3-colourable iff the temporal network just constructed is satisfiable, so NP-hardness of the algebra follows. The algebra is further in NP, since it is a subalgebra of an NP-complete algebra. □

As an example, Figure 6.5 shows the construction in the proof above for the connected two-vertex graph $G_2 = \langle\{1, 2\}, \{\{1, 2\}\}\rangle$.

6.3 Reasoning about Knowledge and Time

Reasoning about time points or time intervals is usually only interesting in the context where something we want to reason about is attached to these temporal entities. For instance, we may want to say that something is known to be true at

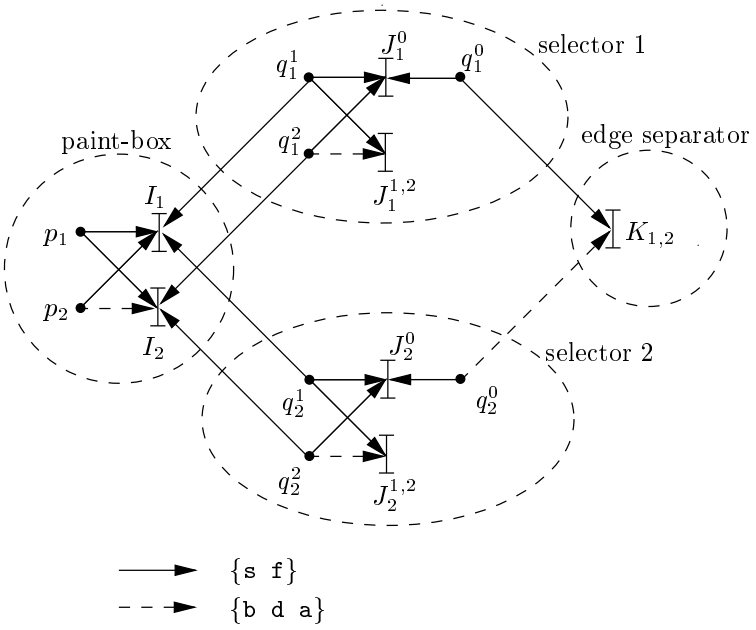

Figure 6.5: An example of the construction in the proof of Theorem 6.2.26 for a connected two-vertex graph.

a certain time point or over a certain time interval, or that an action is executed over a certain time interval. A choice to make here is whether to keep time and other types of knowledge separate or integrated. This choice arises also on several different layers; we may, for instance, use an integrated representation for communicating with the user but separate time from other knowledge internally in the reasoner. Both approaches will be briefly discussed below.

Separating Time From Other Knowledge

One way to build a reasoner for temporally qualified knowledge is to use separate knowledge bases and reasoners for time and other knowledge (propositions, actions or whatever) and let these cooperate. This principle is illustrated in Figure 6.6. In this system, we have a knowledge base (KB) and a time manager (TM), which are interconnected. The KB can represent and reason about logical formulae which are qualified by temporal intervals, indicating when they are true. For instance, the formula $p@I$ means that the proposition p is true over the interval I. However, the KB cannot reason about time, the intervals are just symbols without meaning in the KB. The TM, on the other hand, can only represent and reason about time intervals, it has no 'knowledge' whatsoever of logical formulae. The figure also illustrates how the two reasoners may cooperate. Suppose the KB knows that p holds over the interval I_1 and that q holds over the interval I_2. This does not allow it to draw any conclusions, since it has no knowledge of

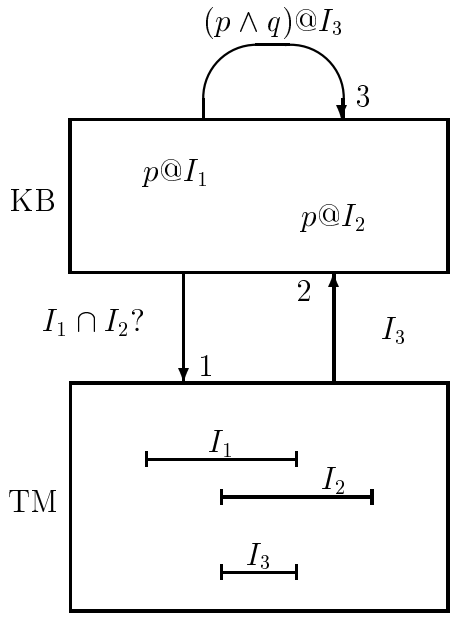

Figure 6.6: Separating reasoning about propositions and time

time or the meaning of I_1 and I_2. If the reasoner now need to know when the conjunction of the formulae p and q is true, it can progress as follows.

1. The KB asks the TM for the maximal interval which is common to the intervals I_1 and I_2.

2. The TM introduces a new interval I_3 and constrains this interval to be exactly the requested interval (or it may find that such an interval already exists) and returns it to the KB. In the case the two intervals do not overlap, the TM may instead say that no such interval exists.

3. The KB now adds to its knowledge the fact that the formula $p \wedge q$ holds over the interval I_3.

For pedagogical reasons, the TM's data base has been illustrated graphically in the figure. In reality it would rather contain interval-algebra formula. That is, initially it would perhaps contain the formula $I_1\{o\}I_2$, and after computing the request from the KB it would also contain the formulae $I_3\{f\}I_1$ and $I_3\{s\}I_2$, or some other equivalent formulae. An example of a separated approach appears in Schwalb *et. al.* [1994].

Integrating Time With Other Knowledge

Another approach is to integrate both facts and time within the same formalism, which is usually done by defining a *temporal logic*, that is a logic which is

augmented to qualify formulae temporally. One generally distinguishes between *tense logics* and *explicit temporal logics*,[11] which will both be briefly described below.

Tense logic

A tense logic is a form of modal logic [Chellas, 1980, Hughes and Cresswell, 1984], with relative time, but no metric time. The usual modalities are F, P, G, H and formulae are interpreted relative to the reference time *now* in the following way:

α means that α is true now

Fα means that α will be true at some future time point

Pα means that α was true at some past time point

Gα means that α will be true at all future time points

Hα means that α was true at all past time point

As an example, let the proposition p denote the claim that there exists a country named Sweden. Then the formula

$$P(H\neg p \ \wedge Fp)$$

denotes the claim that until some time point in the past there never was a country called Sweden and since that time point there has always been and will always be a country called Sweden. An extensive treatment of tense logic can be found in Rescher and Urquhart [1971].

Although basic tense logic has no concept of metric time, there are approaches to adding such capabilities. For instance, one might consider a modification to the P operator s.t. the formula $\mathrm{P}_k\alpha$ says that α was true k time steps ago.

Explicit Temporal Logic

An explicit temporal logic can refer to explicit time points and may or may not define a metric over these, so the previous choice between qualitative and quantitative time prevails also here. There are several common ways to define syntaxes for explicit temporal logics.

An elegant approach to introduce time in a logic is to augment the syntax so that arbitrary formulae can be qualified by time stamps in the following way. For instance, the fact that the formula $P(x) \wedge Q(x, y)$ holds between time points t_1 and t_2 could be expressed as

$$[t_1, t_2](P(x) \wedge Q(x, y)).$$

[11] Many authors use the term temporal logic for tense logic, while other authors use temporal logic as a subsuming term for both tense logic and explicit temporal logic.

This is a syntactically elegant approach favoured by some authors [Shoham, 1987, Sandewall, 1994]. All 'ordinary' formulae must be time stamped—the formula $P(x) \wedge Q(x, y)$ *per se* is meaningless.[12] A temporal formulae must be allowed for expressing temporal relationships, however, *eg.* $t_1 < t_2$, since it would not make sense to qualify such expressions temporally. For instance, the formula $[t_1, t_2](t_1 < t_2)$ could not be given any reasonable semantics. Furthermore, we must add axiom schemata to make the logical connectives take the temporal extent of formulae into account, *eg.* the axiom schema:

$$[t_1, t_2]\alpha \wedge [t_3, t_4]\beta \rightarrow [\max(t_1, t_3), \min(t_2, t_4)](\alpha \wedge \beta)$$

(where axioms for the functions min and max must be added, of course).

Some authors prefer not to step outside the syntax and semantics of ordinary first-order logic, making it necessary to take some other approach. One such approach is to augment all predicates with two extra arguments, denoting the starting and ending time points of temporal interval over which the formula holds. The previous example would then look as follows:

$$P(x, t_1, t_2) \wedge Q(x, y, t_1, t_2).$$

This approach is taken by Bacchus *et. al.* [1991], for example. This has the drawback that we cannot put time stamps on arbitrary formulae, but only on atoms, making the formulae less clear and more awkward to reason with.

A common approach to avoid this problem is to shift the atemporal object-level formulae into terms and introduce a special predicate *Holds* (also commonly called *True*). Examples of this approach appears in McDermott [1982] and Kowalski and Sergot [1986]. The previous example could then look as follows:

$$Holds(t_1, t_2, and(p(x), q(x, y))).$$

This is a so-called *reified logic* (the object-level formulae are reified, *ie.* made into terms). The advantage is that we can qualify non-atomic formulae temporally while staying within ordinary first-order logic. The drawback is that we must introduce axioms to force the terms corresponding to object-level formulae to behave in the expected way. For instance, we have to introduce two constants $true$ and $false$ and axioms for all predicates expressing that they can only take on the values $true$ and $false$, *eg.* we need the axioms

$$\forall x.p(x) = true \vee p(x) = false$$
$$\forall x \forall y.q(x, y) = true \vee q(x, y) = false.$$

In fact, unless we use a second-order logic or axiom schemata we have to add such axioms for all predicates. Similarly, we must add axiom schemata (or whatever)

[12] Of course, it may be given a meaning by the semantics. For instance, it could mean that $P(x) \wedge Q(x, y)$ is always true, *ie.* true at all time points.

for the functions corresponding to logical connectives, *eg.*

$$\forall x \forall y. and(x, y) = true \vee and(x, y) = false$$
$$\forall x \forall y. and(x, y) = true \leftrightarrow x = true \wedge y = true.$$

Just as in the first approach we need to axiomatize the temporal behaviour of the reified logical connectives, adding axioms like

$$\forall x \forall y. Holds(t_1, t_2, x) \ \wedge \ Holds(t_3, t_4, y) \ \rightarrow$$
$$Holds(\max(t_1, t_3), \min(t_2, t_4), and(x, y))$$

(with additional axioms for min and max).

It is easy to introduce quantitative time into these logics. For instance, we can say that the formula α is true for an interval with a duration of at least 17 time units and including the explicit time point 45:

$$[t_1, t_2]\alpha \ \wedge \ (t_2 - t_1) \geq 17 \ \wedge \ t_1 \leq 45 \leq t_2.$$

Further, an explicit temporal logic need not take time points as primitive. Allen [1984] defines a reified temporal logic using intervals as the primitive entity and includes axioms for the interval algebra. However, there are no time points in this logic, so intervals cannot be reencoded as pairs of time points. The fact that the formula $P(x) \wedge Q(x, y)$ holds over the interval I is expressed as

$$Holds(I, and(p(x), q(x, y))).$$

Furthermore, when reasoning not only about truth values, but also about actions or events, it is common to also introduce some predicate *Occurs*, in analogy with *Holds*, to express that a certain even or action takes place over a certain time interval.

Shoham [1987] discussed so-called *hereditary properties* of temporally qualified propositions. For instance, does a proposition that holds over a certain interval also hold over all subintervals and does a proposition that holds at all time points in an interval also hold over the whole interval?

6.4 Temporal-reasoning Systems

Various systems for temporal reasoning have been implemented or suggested. One of the first systems was *TMM (Time Map Manager)* [Dean and McDermott, 1987]. The TMM maintains a network of time points and information about upper and lower bounds for the metric duration between these, thus implementing a metric time-point algebra. It is also possible to state that propositions are true or false at certain time points, and TMM implements a clipping mechanism, that is, if a proposition p is true at a time point t_1 and false at t_2,

where $t_1 < t_2$, then the systems realizes that p must become false at some time point between t_1 and t_2. Various newer and extended versions of TMM exist [Schrag *et al.*, 1992, Boddy, 1993b, Materne and Hertzberg, 1991] and it is being implemented in a commercial version by Honeywell.

Many of the approaches already mentioned earlier have been implemented into temporal-reasoning systems. For instance, systems reasoning about qualitative and metric relations over time points include Tachyon [Stillman *et al.*, 1993], TimeGraph II [Gerevini *et al.*, 1993, Gerevini and Schubert, 1993], IxTeT [Ghallab and Alaoui, 1989], and systems by Barber [1993] and Koubarakis [1992].

Systems reasoning about mixed qualitative and metric information include MATS [Kautz and Ladkin, 1991] and the system used by Dorn [1992, 1994] to reason about intervals and metric time for scheduling.

The IxTeT system has been used as a basis for various other tasks requiring temporal reasoning, including a temporal planner [Ghallab and Laruelle, 1994, Laborie and Ghallab, 1995], a scenario-recognition system [Dousson *et al.*, 1993] and a diagnosis system [Milne and others, 1994]. There are also other planning systems using the TMM or similar time-point reasoners to support temporal planning [Drabble and Kirby, 1991, Rutten and Hertzberg, 1993].

6.5 Further Reading

This section collects a number of references for further reading on various topics extending or being related to the topics of this chapter.

Surveys, Tutorial *etc.*: There is a recent survey/tutorial of temporal reasoning by Vila [1994]. There is also a special issue of the SIGART Bulletin on temporal-reasoning systems, with introduction by Boddy [1993a].

Fuzzy temporal reasoning: Dubois and Prade [1989] describe a fuzzy version of the interval algebra.

Reference hierarchies: There is work on clustering temporal information locally and use reference hierarchies for efficient indexing [Allen, 1983, Dean, 1989, Davis and Carnes, 1990].

Propagation of metric intervals w. uncertainty: Rit [1986] describes an algorithm for propagation intervals with uncertain metric start and end times and uncertain duration.

Cognitively indistinguishable relations: Freksa [1992] investigates interval relations that are hard to distinguish from each other, especially from a cognitive point of view. For instance, in many cases it is impossible to distinguish between the relations before and meets.

Temporal Diagnosis: There is an emerging literature on reasoning about time in diagnosis [Dague *et al.*, 1990, Friedrich and Lackinger, 1991] [Console *et al.*, 1992, Dressler, 1994] [Nejdl and Gamper, 1994].

Bibliography

[AAAI-91, 1991] American Association for Artificial Intelligence. *Proceedings of the 9th (US) National Conference on Artificial Intelligence (AAAI-91)*, Anaheim, CA, USA, Jul 1991. AAAI Press/MIT Press.

[AAAI-96, 1996] American Association for Artificial Intelligence. *Proceedings of the 13th (US) National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, USA, Aug 1996.

[Allen, 1983] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Allen, 1984] James F Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Bacchus *et al.*, 1991] Fahiem Bacchus, Josh Tenenberg, and Johannes A Koomen. A non-reified temporal logic. *Artificial Intelligence*, 52:87–108, 1991.

[Bajcsy, 1993] Ruzena Bajcsy, editor. *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, Chamb'ery, France, Aug–Sep 1993. Morgan Kaufmann.

[Barber, 1993] Federico A. Barber. A metric time-point and duration-based temporal model. *SIGART Bulletin*, 4(3):30–49, 1993.

[Boddy, 1993a] Mark Boddy. AAAI-92 workshop report: Implementing temporal reasoning. *SIGART Bulletin*, 4(3):15–16, 1993.

[Boddy, 1993b] Mark Boddy. Temporal reasoning for planning and scheduling. *SIGART Bulletin*, 4(3):17–20, 1993.

[Bruce, 1972] Bertram C. Bruce. A model for temporal references and its application in a question answering program. *Artificial Intelligence*, 3:1–25, 1972.

[Chellas, 1980] Brian F Chellas. *Modal Logic*. Cambridge University Press, Cambridge, 1980.

[Console *et al.*, 1992] L. Console, L. Portinale, D. Theseider Dupré, and P. Torasso. Diagnostic reasoning across different time points. In Bernd Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, pages 369–373, Vienna, Austria, Aug 1992. Wiley.

[Dague *et al.*, 1990] Philippe Dague, Olivier Jehl, and Patrick Taillebert. An interval propagation and conflict recognition engine for diagnosing continuous dynamic systems. In G Gottlob and W Nejdl, editors, *Expert Systems in Engineering: Principles and Applications. International Workshop*, volume 462 of *Lecture notes in Artificial Intelligence*, pages 16–31, Vienna, Austria, Sep 1990. springer.

[Davis and Carnes, 1990] William S. Davis and James R. Carnes. Clustering temporal intervals to generate reference hierarchies. In *Proceedings of the 8th (US) National Conference on Artificial Intelligence (AAAI-90)*, pages 111–117, Boston, MA, USA, Aug 1990. American Association for Artificial Intelligence, MIT Press.

[Dean and McDermott, 1987] Thomas L Dean and Drew V McDermott. Temporal data base management. *Artificial Intelligence*, 32:1–55, 1987.

[Dean, 1989] Thomas Dean. Using temporal hierarchies to efficiently maintain large temporal databases. *Journal of the ACM*, 36(4):687–718, 1989.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[Dorn, 1992] Jürgen Dorn. Temporal reasoning in sequence graphs. In *Proceedings of the 10th (US) National Conference on Artificial Intelligence (AAAI-92)*, pages 735–740, San Jos'e, CA, USA, Jul 1992. American Association for Artificial Intelligence.

[Dorn, 1994] Jürgen Dorn. Hybrid temporal reasoning. In Anthony G. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, pages 623–629, Amsterdam, Netherlands, Aug 1994. Wiley.

[Dousson *et al.*, 1993] Christophe Dousson, Paul Gaborit, and Malik Ghallab. Situation recognition: Representation and algorithms. In Bajcsy [1993], pages 166–172.

[Drabble and Kirby, 1991] Brian Drabble and Richard Kirby. Associating A.I. planner entities with an underlying time point network. In Hertzberg [1991], pages 27–35.

[Drakengren and Jonsson, 1996a] Thomas Drakengren and Peter Jonsson. Maximal tractable subclasses of Allen's interval algebra: Preliminary report. In AAAI-96 [1996], pages 389–394.

[Drakengren and Jonsson, 1996b] Thomas Drakengren and Peter Jonsson. Quantitative temporal information for maximal tractable subclasses of Allen's interval algebra. Unpublished manuscript, 1996.

[Dressler, 1994] Oskar Dressler. Model-based diagnosis on board: Magellan-MT inside. In *Working Notes of the Fifth International Workshop on Principles of Diagnosis*, New Paltz, NY, USA, 1994.

[Dubois and Prade, 1989] Didier Dubois and Henri Prade. Processing fuzzy temporal knowledge. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):729–744, Aug 1989.

[Freksa, 1992] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54:199–227, 1992.

[Friedrich and Lackinger, 1991] Gerhard Friedrich and Franz Lackinger. Diagnosing temporal misbehavior. In Ray Reiter and John Mylopoulos, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 1116–1122, Sydney, Australia, Aug 1991. Morgan Kaufmann.

[Gerevini and Schubert, 1993] Alfonso Gerevini and Lenhart Schubert. Efficient temporal reasoning through timegraphs. In Bajcsy [1993], pages 648–654.

[Gerevini *et al.*, 1993] Alfonso Gerevini, Lenhart Schubert, and Stephanie Schaeffer. Temporal reasoning in Timegraph I–II. *SIGART Bulletin*, 4(3):21–25, 1993.

[Ghallab and Alaoui, 1989] Malik Ghallab and Amine Mounir Alaoui. Managing efficiently temporal relations through indexed spanning trees. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1297–1303, Detroit, MI, USA, Aug 1989. Morgan Kaufmann.

[Ghallab and Laruelle, 1994] Malik Ghallab and Hervé Laruelle. Representation and control in IxTeT, a temporal planner. In Kristian Hammond, editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS'94)*, pages 61–67, Chicago, IL, USA, jun 1994. AAAI Press.

[Golumbic and Shamir, 1993] Martin Charles Golumbic and Ron Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM*, 40(5):1108–1133, 1993.

[Hertzberg, 1991] Joachim Hertzberg, editor. *European Workshop on Planning*, volume 522 of *Lecture notes in Artificial Intelligence*, Sankt Augustin, Germany, Mar 1991. springer.

[Hughes and Cresswell, 1984] G E Hughes and M J Cresswell. *A Companion to Modal Logic*. Methuen, London, 1984.

[Jonsson and Bäckström, 1996a] Peter Jonsson and Christer Bäckström. A linear-programming approach to temporal reasoning. In AAAI-96 [1996], pages 1235–1240.

[Jonsson and Bäckström, 1996b] Peter Jonsson and Christer Bäckström. Reasoning about disjunctive linear relations. 1996. Unpublished manuscript.

[Jonsson *et al.*, 1996] Peter Jonsson, Thomas Drakengren, and Christer Bäckström. Tractable subclasses of the point-interval algebra: A complete classification. In J. Doyle and L. Aiello, editors, *Proceedings of the 5th International Conference on Principles on Knowledge Representation and Reasoning (KR-96)*, Cambridge, MA, USA, Oct 1996. Morgan Kaufmann.

[Karmarkar, 1984] N Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[Kautz and Ladkin, 1991] Henry Kautz and Peter Ladkin. Integrating metric and temporal qualitatvie temporal reasoning. In AAAI-91 [1991], pages 241–246.

[Khachiyan, 1979] L G Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[Koubarakis, 1992] Manolis Koubarakis. Dense time and temporal constraints with $\neq$. In Swartout and Nebel [1992], pages 24–35.

[Kowalski and Sergot, 1986] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[Laborie and Ghallab, 1995] Philippe Laborie and Malik Ghallab. Planning with sharable resource constraints. In Chris Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1643–1649, Montr'eal, PQ, Canada, Aug 1995. Morgan Kaufmann.

[Materne and Hertzberg, 1991] Stefan Materne and Joachim Hertzberg. MTMM—extending and correcting time map management. In Hertzberg [1991], pages 88–99.

[McDermott, 1982] Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[Meiri, 1991] Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In AAAI-91 [1991], pages 260–267.

[Milne and others, 1994] R. Milne et al. TIGER: real-time situation assessment of dynamic systems. *Intelligent Systems Engineering*, pages 103–124, Autumn 1994.

[Nebel and Bürckert, 1995] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.

[Nejdl and Gamper, 1994] Wolfgang Nejdl and Johann Gamper. Model-based diagnosis with qualitative temporal uncertainty. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI'94)*, 1994.

[Rescher and Urquhart, 1971] Nicholas Rescher and Alasdair Urquhart. *Temporal Logic*. Springer, Vienna, 1971.

[Rit, 1986] Jean-Francois Rit. Propagating temporal constraints for scheduling. In *Proceedings of the 5th (US) National Conference on Artificial Intelligence (AAAI-86)*, pages 383–388, Philadelphia, PA, USA, Aug 1986. American Association for Artificial Intelligence, Morgan Kaufmann.

[Rutten and Hertzberg, 1993] Eric Rutten and Joachim Hertzberg. Temporal planner = nonlinear planner + time map manager. *AI Communications*, 6(1):18–26, Mar 1993.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[Schrag *et al.*, 1992] Robert Schrag, Mark Boddy, and Jim Carciofini. Managing disjunciton for practical temporal reasoning. In Swartout and Nebel [1992], pages 36–46.

[Schwalb *et al.*, 1994] Eddie Schwalb, Kalev Kask, and Rina Dechter. Temporal reasoning with constraints on fluents and events. In *Proceedings of the 12th (US) National Conference on Artificial Intelligence (AAAI-94)*, pages 1067–1072, Seattle, WA, USA, Jul–Aug 1994. American Association for Artificial Intelligence.

[Shoham, 1987] Yoav Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104, 1987.

[Stillman *et al.*, 1993] Jonathan Stillman, Richard Arthur, and Andrew Deitsch. Tachyon: A constraint-based temporal reasoning model and its implementation. *SIGART Bulletin*, 4(3):T1–T4, 1993.

[Swartout and Nebel, 1992] Bill Swartout and Bernhard Nebel, editors. *Proceedings of the 3rd International Conference on Principles on Knowledge Representation and Reasoning (KR-92)*, Cambridge, MA, USA, Oct 1992. Morgan Kaufmann.

[van Beek and Cohen, 1990] Peter van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6(3):132–144, 1990.

[Vila, 1994] Lluís Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, 1994.

[Vilain, 1982] Marc B Vilain. A system for reasoning about time. In *Proceedings of the 2nd (US) National Conference on Artificial Intelligence (AAAI-82)*, pages 197–201, Pittsburgh, PA, USA, Aug 1982. American Association for Artificial Intelligence.