A Causal Reasoning Approach to Behaviour-oriented Design

Keiichi Nakata

PhD Thesis
Department of Artificial Intelligence
The University of Edinburgh

1994

A Causal Reasoning Approach to Behaviour-oriented Design

Keiichi Nakata

PhD Thesis
Department of Artificial Intelligence
The University of Edinburgh

1994

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Keiichi Nakata
Edinburgh
30 May 1994

Abstract

This thesis describes a basis for applying causal reasoning to the problem domain of design. Causal reasoning covers one of the fundamental aspects of reasoning humans perform when dealing with dynamic change. Such temporal features prevail in design problems which aim to provide artefacts based on some desired behaviour. This thesis first investigates causal reasoning to develop a language with expressiveness for representation and reasoning about change and for application to design. The second half of the thesis describes how this can be used for simple design tasks.

The language for causal reasoning is based on the logic CI introduced by Yoav Shoham. CI is a nonmonotonic temporal logic for prediction, which prefers those models which are chronologically more ignorant, i.e., if changes were to take place they should happen as late as possible. It was presented as a method for dealing with frame problems. In this thesis, a framework for abduction is suggested for CI. It assumes that certain states persist not only forward, but also backward. This thesis argues that it is often the case that reasoning about the past events is in fact reasoning about the events between two or more time points. Problems of such nature are referred to as interpolation problems, and this thesis describes a way of dealing with this class of problems by bidirectional sweeping, which performs reasoning forwards and backwards over the time range. The outcome of this operation is a plausible sequence of events that took place in that time range.

This method of causal reasoning is applied to design problems; by focusing on how devices work, we develop the notion of behaviour oriented design which aims to achieve the desired behaviour of devices and their interactions with their surroundings. Essentially, the specification is provided as the sequences of events which are desired to happen by functioning of the device in the given environment. All the design knowledge involved is represented by causal theories of the language CI, which provides a mechanism to predict the performance of the current design. The comparison between the behavioural specification and the predicted model allows verification of designs and suggestions for modifications to them. The proposed design framework is useful for domains which involve dynamic change and in which design requirements can readily be expressed as temporal sequences of events.

Acknowledgements

There is a number of individuals and organisations to whom I am indebted. I can only mention only a small portion of them in this limited space which otherwise may compose an extra volume.

I first would like to thank my supervisors, Dave Robertson and Alan Smaill, for invaluable advice, inspiring discussions, enlightenment, and most of all, encouragement in the process of the completion of this thesis. Alan's profound knowledge of logic and Dave's expertise in knowledge based systems enabled me to work on the topic that bridges between the two fields.

I also would like to thank Murata Overseas Scholarship Foundation for providing full financial assistance for the first two years of my research. I was also supported by Overseas Research Students Awards Scheme, reference ORS/9214048. In addition, I would also like to acknowledge the Department of Artificial Intelligence for providing fundings for travels to conferences and workshops where I was able to present parts of this work.

My gratitude extends to my colleagues in the Department of Artificial Intelligence for moral support during my unforgettable time in Edinburgh. Among them, I would specially like to mention the names of fellow students, Alan Black, Rolando Carrera-Sanchez, Flávio Corrêa da Silva, Matt Crocker, Carla Pedro Gomes, Ian Frank, Khee Yin How, Alistair Knott, Ian Lewin, Nelson Ludlow, Suresh Manandhar, Wamberto Vasconcelos, Rob Scott, and Maria Vargas-Vera, for the time we shared in E17 South Bridge. I also would like to thank Professor Shunsuke Kondo, Professor Kazuo Furuta and Keisuke Sato of the University of Tokyo for their encouragement towards my study in Scotland. Professor Mark Lee, Andy Ormsby and Patrick Olivier of the Centre for Intelligent Systems, University of Wales, Aberystwyth were among those who supported me with understanding in the later stage of this work.

Finally, I would like to express my sincere gratitude to my parents to whom I dedicate this thesis.

Contents

List of Figures

List of Tables

Notations and Abbreviations

List of notations

$\Box$	'necessary' operator
$\Diamond$	'contingent' operator
$\supset$	causal implication
$\Leftrightarrow$	biconditional implication
$\Rightarrow$	material implication
$A \vdash B$	B can be proved from A
$A \models B$	A is a model of B

List of abbreviations

CI	chronological ignorance
ltp	latest time point
etp	earliest time point
cmi	chronologically maximally ignorant
bci	chronologically ignorant backwards

Chapter 1

Introduction

Imagine a medium sized lecture theatre which has two entries on the opposing sides. Students can enter and leave the lecture theatre from either entry, depending on where they come from and where they are going to after the lecture. It has no window to allow light in, so the first one to come into the place would turn on the light by operating the switch by the door through which she came in, and the last one to leave would turn off the light, again by operating the switch by the door though which he is walking off. It is typically the case that the door from which the first student came in and the door from which last student went out do not need to be the same one in order for this sequence of switch operations to do the intended job. This is a very familiar setup and there is a well established way of implementing it. The problem is, if we had never seen such a useful system, how would we go about designing it?

To begin with, we should analyse how these switches are operating. Let's start by naming the two entries Entry1 and Entry2. When someone operates the switch at Entry1, it turns the light on. Then if someone operates the switch at Entry2, it turns the light off. But then if someone operates the switch at Entry1, then the light comes back again. What is being studied here is the behaviour of the system, i.e., how the system performs over a sequence of events. There are several such behaviours to this system, and the goal of the design is to construct a device which exhibits all those behaviours. These behaviours are typically produced by causal chains, governed by underlying causal theories. The device can be seen as an artefact which brings about

such causal relations in the world (Figure 1.1). The motivation for designing the device is to provide the desired behaviours.

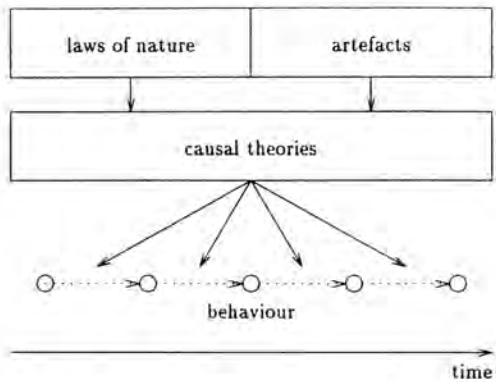


Figure 1.1: Relation between artefacts, causal theories and behaviours.

The work presented in this thesis aims to cover the fundamentals of such reasoning and its role in design. Causal reasoning is considered to play an important part in the reasoning process. How we can represent design knowledge in the framework of causal reasoning is another important element of this task.

1.1 Outline

The contents and the flow of arguments presented in the following chapters of this thesis can be outlined in the following way.

1. Causal reasoning is one of the fundamental reasoning methods performed by humans when thinking about change. It is based on the notion of causation which identifies the connection between two or more events distinguished as cause and effect. Causation has various properties among which are nonmonotonicity, asymmetricity, and the involvement of temporal dimensions. Most important of

all, the causal analysis of a domain gives rise to heuristics to reason efficiently about change with incomplete information. Given these properties, causal reasoning may be applied for problem solving in a domain where reasoning efficiently about change is crucial.

2. Although there are various possible worlds which are accessible by means of causal rules, it is useful to have a notion of 'most likely' outcome. In a prediction problem, it is often computationally very complex to maintain all possibilities, and this leads to notorious frame problems. As a way of handling this complexity, Shoham devised a logic of chronological ignorance (CI) based on the preference criterion which prefers the model where a state persists (or continues to be true) as long as possible. Given a sound and consistent causal theory, the logic CI constructs a unique model. Shoham's propositional framework has been extended with first-order features, to allow more expressiveness in the causal theory.

3. It is suggested that the notion of persistence can be applied not only forward in time, but backward. If we have a sound causal theory, we can reason about what should have happened in the past given the present events by abduction. Procedurally, this is very much like reasoning about the future, but applying the rule backwards. However, in most cases reasoning about the past is bound by boundary conditions. That is, it is often the case that reasoning about the past events is in fact reasoning about the events between two or more time points. An extension to Shoham's framework based on backward persistence enhances its capability to reasoning about past events.

4. The major areas of application for causal reasoning so far have been prediction and, more recently, diagnosis. Its application to design has not been well explored. A logic model of design is proposed based on causal reasoning framework. For this purpose, a formal model of design is investigated, which is useful in generalising design methods.

5. In the causal reasoning framework, the focus is set on the events and their relations. That is, the sequence of events is emphasised rather than physical features. It then naturally follows that specifying a design in terms of its performance over

time, or its *behaviour*, is possible and enables us to capture some functional aspects of a device. The causal theory provides a mechanism to predict the performance of the current design based on the model constructed by the nonmonotonic temporal framework devised earlier. The comparison between the behavioural specification and the predicted model can be considered at the verification phase of design.

6. The purpose of design now is to construct a causal theory from which the desired behaviour can be obtained. Causal rules are added, retracted or modified based on the occurrence of necessary/undesirable events. Causal abduction identifies what events are necessary to obtain desired behaviour and provides a pointer to construct causal rules which are necessary to obtain such events.

7. The proposed design framework is useful for domains which involve dynamic change and where temporal sequence is important to performance. It is tested on sample problems, mainly in the domain of logic circuits, and application to other domains such as process engineering is expected.

1.2 Motivation and aims

The research described in this dissertation is motivated by the following problems and is aimed to achieve the tasks indicated.

Enhancement of the expressiveness of Shoham's temporal logic. Shoham[88] suggested a nonmonotonic temporal language which is aimed to deal with the frame problem. This language is based on the notion of *chronological minimisation*, which minimises the number of events in that if changes occur they take place as late as possible. By implementing this language in both propositional and first-order cases, this thesis suggests ways of representing various problems in causal and temporal reasoning.

Reasoning about the future and the past in the same framework. Although reasoning about change involves reasoning about the future as well as that of the past,

most causal reasoning frameworks only deal with one of these. Since both prediction and reconstruction—reasoning about what events took place in the past—are based on the knowledge of causation, it is useful to share the same representation and formulation of causation. A language to handle reconstruction tasks, based upon Shoham's temporal language, is desired for this purpose.

Application of causal reasoning in a practical domain. The human behaviour of designing is one of the most complex tasks involving various types of knowledge and reasoning methods. Causal reasoning in design is one of the important aspects of design which provides access to the modelling of dynamic processes and temporal behaviours of devices. I aim to apply the temporal language developed above to represent causal knowledge and causal reasoning.

1.3 Readers' guide to this dissertation

The following diagram shows the dependency between the chapters in the thesis (Figure 1.2). Chapters 2,3 and 4 introduce and discuss the language we use to represent and manipulate causal knowledge. The second half of the dissertation, which consists of chapters 5,6,7 and 8, is devoted to the application of causal reasoning to design problems. Readers not interested in formal descriptions of logic may skip chapters 3 and 4 and proceed to chapter 5.

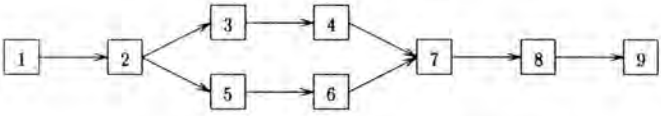


Figure 1.2: The structure of the thesis.

Chapter 2

Causation and causal reasoning

This chapter serves as an introduction to the accounts of causation and the elements of causal reasoning. Relations which are conceived as causation form a useful notion in reasoning about events and changes, but this has been one of the most controversial topics in philosophy. Although the essence of causation is yet to be understood formally, there is no doubt that we make predictions and come up with explanations based on some knowledge about causation. We first take a look at the historical accounts of causation and causal reasoning and provide philosophical backgrounds of causation. This survey is by no means complete, given that causation has been one of the most discussed topic in philosophy since Aristotle up to the present, but it is useful to give philosophical grounds to the treatment of causation in this thesis. Some technicalities related to the treatment of causation in the field of artificial intelligence (AI) are described in the latter half of the chapter.

2.1 Role of causation in reasoning

Let us examine the properties of causation and what are perceived as causation. Since causation has been a scope of philosophical investigations throughout history and the topic itself would produce volumes of books, it is best to focus on a particular account of causation. We examine here Shoham's account of causation and explore his language of representing and reasoning about causation (Chapter 3). Shoham created a partial list consisting of eight properties of causation which can be summarised as

follows [Shoham 88, pp.152–155].

1. Causation is different from material implication.
2. Causation is nonmonotonic.
3. Causation is indexical.
4. Causation is asymmetric and antireflexive.
5. Cause has temporal precedence over effect.
6. Causation has temporal dimension.
7. Causation can both enable and prevent events.
8. Causation should be computational.

Causation differs from material implication since there should be a notion of 'causing' involved between antecedents and consequents—it is quite clear that the verbs "implies" and "causes" are not interchangeable. Causation being nonmonotonic is connected with its being indexical (context-sensitive). Turning the electric heater on causes the room to become warm, but not if the electricity is cut off. Most of the causal relations we perceive are usually under a particular environment, or "causal field" as described by John Anderson[38] that there is some underlying domain in general causal statements.

Asymmetry of causation prevents circular causation. Antireflexivity prevents an event from causing itself.[1] Our intuition tells us that cause precedes effect in time, but some philosophers talk about simultaneous causation, or even backward causation.[2] It is evident that a temporal dimension must be involved if we are to discuss temporal precedence.

'Enable' and 'prevent' are two verbs which describe some specific aspects of 'causing' an event. Shoham points out that 'p prevented q' is not equivalent to 'p caused $\neg q$'. This

[1] Apart from, perhaps, God.

[2] Since backward causation steps into the field of ethics or teleology, I want to avoid the discussion on this topic, but simultaneous causation is discussed later in Section 2.2.

subtle difference can be illustrated by the following example: 'The emergency import of food prevented starvation' implies that starvation was expected if the emergency import of food did not take place; whereas 'the emergency import of food caused not-starvation' does not carry the same implication. A similar difference exists between the verbs 'enabled' and 'caused'.

The requirement that causation should be computational is emphasised by Shoham as the "foundation of [his] own account of the concept". This indicates his motivation of constructing a language for causation which is computational, and the logic he devised fulfills all the eight properties (See Chapter 3). The language with such capability to cover these aspects of causation is indeed very attractive and worth investigating. Furthermore, it seems useful to explore its applicability to a range of problems concerning causation.

Before we focus on Shoham's logic, let us skim through the analyses of causation by various philosophers and the concept of causation in physics which is closely related to the practical application of causal reasoning.

2.1.1 Philosophical standpoint

As mentioned earlier, a survey of investigations into causation can never be complete and is far beyond the scope of this dissertation. However, it is useful to have an idea of what some of the philosophers have suggested to place Shoham's list in them. Shoham himself chose three modern proposals, those by Mackie, Lewis and Suppes [Shoham 88]. Among these, Mackie's account seems most relevant to Shoham's proposal, and it is listed below among short paragraphs describing those of earlier investigations by David Hume and J. S. Mill.

David Hume One of the most influential investigations into causation was that by David Hume[1739]. Hume suggested three criteria for the relation between cause and effect: spatio-temporal contiguity, temporal priority of the cause, and necessary connection between cause and effect. Hume placed his emphasis on the last criterion which rendered 'causal necessity', and stressed the importance of distinguishing causal

necessity and logical necessity.

J. S. Mill Later, J. S. Mill[1843] suggested that it is important to distinguish between a cause and a condition. Mill's point was that causation is not a relation between a consequent and an antecedent, but between a consequent and the sum of several antecedents.[3] The cause might be one or more of those antecedents with the others being conditions, or necessary conditions, that enable the cause to invoke the effect.

J. L. Mackie Mackie[74] suggested INUS condition—an *insufficient* but *nonredundant* part of an *unnecessary* but *sufficient* condition. Given a conjunctive formula $P \wedge Q \wedge R$ which represents a condition for S to occur, $P \wedge Q \wedge R$ is an INUS condition if it is a minimal sufficient condition such that none of the conjuncts, P, Q, R, is redundant. Other such definitions based on necessity/sufficiency without referring to temporal constraints, however, have the danger of not distinguishing the cause and effect by relying too much on mutual dependence. It must be emphasised that temporal sequence cannot be ignored in identifying causal relations.

Just comparing the criteria of causation by Hume and the properties in Shoham's list, nothing is said in the latter about the spatio-temporal contiguity and the causal necessity. The closest the list gets to these concepts is to point out that causation is context-sensitive and has a temporal dimension, but this is not quite the same as referring to spatio-temporal contiguity nor causal necessity. The difference between Shoham and Hume is the way they approach causation. Hume's approach to causation, as well as many other philosophers', is to identify what can or cannot be conceived as causation, while that of Shoham is to list the properties of what are already conceived as causation. The significance of Shoham's analysis is to help create a sound notation to represent causal knowledge without violating the 'form' of causal rules; it is up to those writing causal rules to ensure what is called 'causal necessity'.

[3]Causes treated by Hume were single events/objects.

2.1.2 Physical standpoint

Russell argued that in modern physics the notion of a cause has been replaced by the notion of functional relations [Russell 63]. That is, the aim of a physicist is to formulate laws of nature which enable us to deduce the state of a system at any other time from any given state. This means that explanations in physics are not produced from the analyses of causal chains but by deduction from appropriate physical laws. A similar analysis of physical laws is presented by Feynman[67]. According to Feynman, the physicists' main concern is to say, 'These are the conditions, now what happens next?'; telling about the past from the present is not their usual concern [p.114]. This is unlike geologists or historians whose interest lies in the past. However, by using their theories about the domain, they can just as well 'predict' the possible findings in the future given what has been found.

Despite Russell's claim, attempts to find causal relations in the domain of physics are prevalent. Physical laws may represent functional relations, but nothing can be predicted unless certain parameters are specified to make use of the physical equations. For instance, Newton's law $F = ma$ where F is force, m is the mass of an object and a its acceleration, F can be the cause for the object having the acceleration a, or, having a can be the cause for the object to have the force F. While this is a effective causal analysis, the physical equation itself does not explicitly identify the cause and effect. In this case, what can be considered as *cause* can be seen as what is *known*. This is pointed out by Shoham in his criticism against causal ordering theory [Iwasaki & Simon 86a] that it is "information theoretic dependence" rather than causation.[4] Furthermore, strictly speaking, this relation does not comply with the property 5 of Shoham's list, since there should not be any temporal differences among the force and the acceleration. However, there is a temporal precedence for the force, as a cause, to be perceived by physicists before acceleration is calculated by Newton's law. It will in turn have an opposite temporal precedence when calculating the force from acceleration. There is a temporal precedence when the law is actually applied.

[4]Those interested in the discussions concerning the difference between the concepts of causation among Shoham, Iwasaki & Simon, and deKleer & Brown are suggested to follow their exchange of criticisms in [Simon 91], [Shoham 91], [Iwasaki & Simon 86b] and [deKleer & Brown 86].

It seems that causation is essentially a notion utilised by humans to predict the future and to explain the past in an efficient manner. It is comparable to heuristic rules used in expert systems, which may not have any support but provide efficient and plausible reasoning. This way of looking at rules in general from the point of view of efficiency is strikingly similar to the necessity for nonmonotonic reasoning. There are always exceptions depending on the context where the rules are applied, but it is more efficient to make the most plausible inference based on incomplete information. Similarly, when precise identification of causal chain is time consuming or in some cases impossible, it is reasonable to conceive the relation of events to be causal. This view supports the computational aspect of causation advocated by Shoham, that prediction is feasible only when it can be done efficiently and reliably [Shoham 88, p.162]. At the same time, computability should also be a feature of reconstruction.

It is evident from the investigations into causation by the thinkers of the past that their main interest in applying causation was to answer the 'why' question, or *explanation*, for a given observation. According to Russell, physicists rather than philosophers were the ones who placed more weight on prediction. While Shoham commits himself to the use of causation for prediction, and suggests that the causal knowledge for explanation tasks should have its own form, I find it consistent to apply Shoham's formulation for explanation tasks as well as for prediction. There are two main reasons for this. The first is that not all causal knowledge can be represented as physical equations. It is reasonable to choose a representation which manages the form of causation as relation between cause and effect. The second is that I believe it is more consistent to manage causal knowledge in a uniform manner rather than to prepare causal knowledge of the same domain separately for prediction and explanation.

To preserve the rule form as it is in causal rules, there are only three possible ways to relate cause and effect as antecedent and consequent. The first is to make sure that the cause, as the antecedent, always precedes the effect which is the consequent. The second is the transposition of the first, to have the effect as the antecedent and the cause as the consequent. And the third is to allow simultaneous causation, and for that matter allow the temporal priority between the cause and the effect. Among the three, the

first complies with the condition set by Shoham and quite obviously is the reasonable choice. The second case would represent 'backward causation', the notion proposed by some philosophers (See [Dummet 93] for example) to describe pre-emptive action, but this concept is rather psychological and teleological which is beyond our scope. Using the uniform representation, describing causation by the form '*if* an event E occurred *then* C must have happened before E' is unsound and rather counter-intuitive. The third can be handled by the first case by extending the notion of temporal ordering in the following section. This leaves the first formulation to be the most plausible: and this choice supports the application of Shoham's causal theory to reasoning about the past which is discussed extensively in Chapter 4.

2.2 Direction of causation

One of the features of causation according to Shoham was that cause should always precede the effect. Leaving 'backward causation' aside, this assumption is not always supported.[5] The leaden ball example by Kant[1781] illustrates that causation may not necessarily have direction forward in time, but can well be simultaneous. If you take a leaden ball and place it on a cushion as a cause, it creates a hollow as its effect. But these occur simultaneously since there is no time between when the leaden ball is placed and the formation of the hollow. In order to explain this example, he distinguishes the *ordering* of time and the *lapse* of time, and claims that the former is a relation and does not necessary involve time.

Shoham[88] admits that there are some cases where simultaneity of cause and effect is natural. However, assuming simultaneous causation may result in circular causation and other technical difficulties such as spatio-temporal conflict, *e.g.* two objects occupying the same space at the same time. Shoham presents no concrete solution to this problem, but I think his intuition is correct. Some of the relationships between parameters in physical equations show that certain parameters change simultaneously. But the issue is why they should change in the first place. There should be a cause for

[5] For instance, Lewis[73] rejects such assumption since it rejects, without any evaluation, even valid physical hypotheses which may pose backward of simultaneous causation.

such change and at least one of those parameters should be responsible for propagating change. In this sense, there is a cause and it is more intuitive to assume causal ordering by means of infinitesimal temporal priority, or perhaps, the ordering of time that Kant supposes.[6]

The identification of cause in the way pictured above is influenced by what is sought for in each problem. It then drifts further away from an objective view of causation and consequently becomes subjective. It may well become the problem of who perceives causal relation for what purpose. This view of causation is congruous with the concept that causation is essentially a useful description of regularities to reason efficiently about occurrence of events as previously discussed.

2.3 Abduction and causation

Owing to the feature of causality that a cause can determine the effect but not vice-versa (an argument opposing 'backward causation'), adopting the logical notation will place causes in the antecedent and effects in the consequent of a causal rule. Moreover, because of the assumption that causes always have temporal priority over effects, applying causal rules to find the effect given an event should always involve a temporal step forward. On the other hand, if we were to use this form of causal rules to explain the past from the present, we need to find a reasonable cause to an effect. Finding a reasonable antecedent from a consequent is known as *abduction*. This form of inference is gaining increasing attention in artificial intelligence [Pople 77, Reggia *et al* 83, Charniak & McDermott 85].

The form of abduction as first described by C. S. Peirce is as follows:

> The surprising fact, C, is observed.
> But if A were true, C would be a matter of course.
> Hence, there is reason to suspect that A is true.
> [Peirce 55, p.151].

[6] This problem is discussed further in Section 3.8.1.

The idea behind using abduction is summarised by Curd[80] into the following three notions.

- Explanatory hypotheses are formed by abduction and only by this process new ideas can be introduced by logical operations.
- Hypotheses are formed by guesses and insights which are intuitive and instinctive.
- Hypotheses formed by abduction are adopted 'on probation', which is to say that hypotheses formed are not accepted as true or probable as in induction, but are regarded as suggestions or working conjectures which are reasonable enough to pursue.

In support of Peirce, philosophers such as Hanson[58] advocate the notion that unlike deduction or induction, abduction can originate ideas and that all scientific ideas are produced by abduction. The examples given are the discovery of Mars' orbit by Kepler and the discovery that gravitational acceleration is constant by Galileo.

In the purest form of abduction, perhaps the hypotheses are provided by insight or some kind of intuition which may not necessarily be explainable. However, in practice, there should be some procedures which are applied to suggest reasonable guesses. Such procedures should be based on certain selection criteria applied to a range of candidates obtained by abductive operation on conditional statements.

Now to place abduction in the context of causal reasoning, let us consider where it fits into the picture. Tasks we consider here are *prediction* and *retrodiction*.[7] A prediction task is to find what is true in the future given present conditions. Conversely, a retrodiction task is to find what was true in the past given present conditions. These two can be formally represented as follows. Here, $\phi(t)$ and $\psi(t)$ are descriptions of system at time t.

Prediction Given $T, \phi(t_0), t_f > t_0$, find ψ such that $T, \phi(t_0) \vdash \psi(t_f)$.

[7] A well described phrase about this term is by Sir Isaiah Berlin, who wrote, "retrodiction—filling in gaps in the past for which no direct testimony exists with the aid of extrapolation performed according to relevant rules or laws." [Berlin 78, p.110]

Retrodiction Given T, $\phi(t_0)$, $t_p < t_0$, find ψ such that $T, \phi(t_0) \vdash \psi(t_p)$.

Notice that both of these are deductive operations. In order to perform these operations, prediction tasks need a causal theory with causal rules of the form *if cause then effect*, while retrodiction requires those of the form *if effect then cause*. This means that these two tasks cannot be done using the same causal theory.

If we accept the notion that causal rules of the form *if cause then effect* are more intuitive than their counterparts, we need *causal abduction* in order to make use of such a causal theory for reasoning about past events. This operation can be represented as:

Causal abduction Given T, $\phi(t_0)$, $t_p < t_0$, find ψ such that $T, \psi(t_p) \vdash \phi(t_0)$.

This is in fact the reverse operation of deduction for prediction. Retrodiction and causal abduction both share the same purpose of reasoning about the past but with different sets of causal theories.

2.4 Possible worlds and causation

When we try to predict the future or explain what had happened in the past based on what we know about the present, there is no single trajectory of events which we can undoubtedly draw. Usually there are many possibilities as to what will happen in the future and what should have happened in the past. It depends on how much information we have about the relevant events. What we want are plausible predictions or explanations. This section provides a brief overview of the concepts and techniques which are involved in the formal description and treatment of causation.

2.4.1 Temporal and modal logics

Since the frame problem [McCarthy & Hayes 69, McCarthy 80] arises in the context of prediction or reasoning about actions and changes, it is clear that a mechanism to handle temporal information is necessary. Temporal logic is a useful tool that is

capable of handling such information. Details of the *interval based* and *time point based* temporal logics are described in the next chapter.

Another tool we need is something that is capable of reasoning about multiple worlds. When we need to take into account and maintain all possible situations, modal logic is a powerful tool. Essentially, modal logic is constructed on top of standard logics such as propositional and predicate logics with the modal operators $\Box$ ("necessary" operator) and $\Diamond$ ("contingent" or "possible" operator). The semantic concept of modal logic can be illustrated by the *Kripke structures* described below. Modal logic is interpreted over this structure of possible worlds.

2.4.2 Kripke structure

A Kripke structure is made up of *possible worlds*; usually the interpretation (by standard logics) is different for each world. The point of modal logic, then, is to identify that a formula is true or not in *a particular world*. If a formula φ is true in *all* possible worlds, then $\Box\varphi$ is true in that Kripke structure. If it is true in *some* worlds, then $\Diamond\varphi$ is true in the Kripke structure.[8]

2.5 Nonmonotonic logics

Nonmonotonicity is another issue in reasoning about change. The difference between *monotonic* logic and *nonmonotonic* logic is that, in the former, the addition of new premises (axioms) will not invalidate old conclusions (theorems) while that is not necessarily true in the latter [Lukaszewicz 90]. When reasoning about change, it is often the case that the reasoning is performed under incomplete information and the previously concluded outcome may need to be retracted in favour of the new conclusion when new information arises.

Several techniques have been developed to deal with nonmonotonicity. Among those, the *closed world assumption* [Reiter 78], *circumscription* [McCarthy 80, McCarthy 86],

[8] Formally, $\Box\varphi$ [$\Diamond\varphi$]is true in a world w iff φ is true in all[some] worlds that are *accessible* from w. ([Hughes & Cresswell 72])

default logic [Reiter 80] and *autoepistemic logic* [Moore 85] which are relevant to Shoham's formulation, are briefly summarised in the following subsections.

2.5.1 Closed world assumption

Closed world assumption assumes that all the necessary positive information is available. Formally, it is described that for any n ary relation symbol R and any n tuple of ground terms $\alpha_1, ..., \alpha_n$, we may assume $\neg R(\alpha_1, ..., \alpha_n)$ unless the contrary can be proved.

It enables us to only represent explicitly the positive information which is true, and it will implicitly infer that those that is not represented are false. It simplifies the representation on one hand, but the information about negative facts is relatively weak. Adding new information may replace the previous one and by this means it can model nonmonotonic reasoning. However, if we were to make a closed world system more flexible, without overriding the information itself and yet deal with nonmonotonicity, formal methods such as circumscription are necessary to do the job.

2.5.2 Circumscription

One of the problems which is apparent in a closed world assumption is that while it can talk about instances and show if they can be proved, it is difficult to talk about a class of entities with exceptions. *Circumscription* is a procedure that treats a first-order theory so that the only instances which satisfy a predicate are those which are known to satisfy it. In other words, the instances of a predicate are limited to those which can be shown to satisfy the predicate.

For example, if there is only one known fact about an object that flies which was *flies(robin)*, by circumscribing the predicate *flies* we can obtain the formula $\forall x$ *flies*$(x) \Leftrightarrow x =$ *robin*. This restricts the instance of the predicate *flies* to *robin*. If we query *flies(canary)*, it is false since we have yet to show that canaries fly. If we then add *flies(canary)* as new knowledge, by the circumscription of *flies* we obtain $\forall x$ *flies*$(x) \Leftrightarrow x =$ *robin* $\vee$ *canary*.

The circumscription is formally expressed as follows [McCarthy 80]. Let A be a first-order sentence containing a predicate P, and $A(\alpha)$ to stand for the result of replacing every occurrence of P in A by a formula α. The circumscription of P in $A(\alpha)$ is the schema of second-order

$$A(\alpha) \wedge \forall x(\alpha(x) \Rightarrow P(x)) \Rightarrow \forall x(P(x) \Rightarrow \alpha(x))$$

where x can be a tuple of variables.

The advantage of using circumscription becomes more apparent when we circumscribe on exceptions [McCarthy 86]. For example, the sentence *all birds that are normal fly*[9] can be formulated as

$$\forall x\, bird(x) \wedge \neg ab(x) \Rightarrow flies(x)$$

where the predicate $ab(x)$ means that x is an abnormal instance of a *bird*. Now if there is knowledge that an ostrich cannot fly and is therefore abnormal, *ab(ostrich)* then by circumscribing on *ab* we can obtain

$$\forall x\, ab(x) \Leftrightarrow x = ostrich$$

Since the only instance that can be proved to be abnormal is an *ostrich*, the first formula can be interpreted as "all birds can fly except ostriches". Obviously the exceptions can be accumulated by proving their abnormality and circumscribing on *ab*. McCarthy gave definition in terms of *minimal models* which prefer models whose pred icate has a smaller extension. Essentially circumscribing a predicate is *minimising* its extension. Shoham's logic described in the next chapter can be taken as a variation of this idea.

2.5.3 Default logic

In *default logic*, the rules are represented by an expression called *defaults*. For instance, the rule in the example above that "birds can fly" is expressed by a default

$$\frac{bird(x) : flies(x)}{flies(x)}$$

[9] This sentence itself implies that the birds that do not fly are abnormal

This default is interpreted as "for every individual x, if x is a bird and it is consistent that x can fly, then x can fly." This means that if x is a bird and as long as it is consistent in the theory that x can fly, then it can fly. If the theory includes that an ostrich is a bird but cannot fly, then it is inconsistent to believe that an ostrich can fly, blocking the application of the default.

The general form of a default is

$$\frac{\alpha(x) : \beta(x)}{\gamma(x)}$$

where $\alpha(x), \beta(x), \gamma(x)$ are called the prerequisite, justification and the consequent of the default, respectively. x can be a tuple of variables [Reiter 80].

Inference using defaults is performed by constructing a *default theory*, which is a pair $T = (W, D)$, where W is a set of first order sentences which are the axioms in T, and D being a set of defaults. It can be seen as a classical theory W being enhanced by defaults. The result of applying defaults to W is an *extension*, and a default theory would have a number of extensions, or no extension. For instance, if there are two defaults in D, $\alpha_1(x) : \beta(x)/\beta(x)$ and $\alpha_2(x) : \neg\beta(x)/\neg\beta(x)$, there will be two extensions since the two defaults block each other.

It is worth noticing that while default logic is expressive and has a nice feature of dealing with multiple extensions, some of them can be counterintuitive and unreasonable. It offers a good representation of default knowledge but requires careful selection of the results.

2.5.4 Autoepistemic logic

The motivation for autoepistemic logic is to model the reasoning of an agent which reflects its own beliefs [Moore 85]. It is a logic augmented by a modal operator L, which stands for "is believed". The default inference is performed by obtaining the stable expansions of an autoepistemic theory, i.e., one closed under the operation

$$Aut(A) = A \cup \{Lp : p \in A\} \cup \{\neg Lp : p \notin A\}.$$

Since an autoepistemic theory treats the belief of the reasoning agent, a typical rule

such as $bird \wedge \neg ab \Rightarrow flies$ is translated as $bird \wedge \neg Lab \Rightarrow flies$ to represent the default to be "it is not believed to be abnormal." In this case, since $\neg Lab$ is included in the autoepistemic expansion, it is not the case that ab is true, therefore given *bird*, *flies* should be true.

The main feature of autoepistemic logic is that it represents the belief of the reasoning agent rather than just defaults. Also as the result of introduction of a belief operator, it brought the modal logic into the area of nonmonotonic reasoning. However, it can allow inconsistencies among defaults and the autoepistemic theory itself can be unsound.

2.6 Causal reasoning in application

Causal reasoning has been applied to various fields of problems apart from pure prediction and reconstruction problems. It is used in the context in which the consideration of causal relations and causal knowledge are advantageous within its domain.

Database management. A temporal database management system (TDBMS) is used for databases which involves temporal information especially in the form of a set of events and their consequents together with its order, duration and time of occurrence. One of the notable works in this domain is TMM [Dean & McDermott 87, Dean & Boddy 87] which introduced time tokens to identify the intervals during which a fact or event is known to be true. By using data dependencies, a fact can be properly placed within the proper temporal relations with respect to other facts. Causal theories specify the causal relationships between events and processes. The significance of TMM is that it can handle partially ordered events with computational efficiency.

Planning. A program that emphasised the applicability of causal reasoning in planning is the SIPE planning system [Wilkins 87]. SIPE is based on the STRIPS assumption that the truth values of predicates do not change without the explicit operation of add or delete lists by events. The significance of SIPE is that it separated knowledge about causation from knowledge about actions and made the representation and

planning process simpler and more expressive. Other works include an attempt to realise Shoham's suggestion that his temporal logic can be applied to planning problems [Bell 91].

Diagnosis. The earliest application of causal reasoning can be said to be in the medical diagnosis represented by MYCIN [Shortliffe 76]. Some of the diagnostic rules MYCIN used were causal rules which were used to identify the cause (disease) from the effect (symptoms). However, the reasoning involved in MYCIN was not strictly causal in the sense that it did not place importance on the sequence of events, or put emphasis on temporal relationship among events. Diagnoses of behaviours of devices are gaining increasing attention (for example [Davis 84]), including a work coupling model-based reasoning and temporal analyses [Friedrich & Lackinger 91]. There is, however, little emphasis on causation in such work and it is seldom referred to as causal reasoning.

Design. The application of causal reasoning is not so significant as of yet in design domain. However, the representation of purpose in design is gaining increasing attention in design verification task. One of the notable works is that by Iwasaki and Chandrasekaran[92] which bases the identification of causal dependence among parameters on the causal ordering theory [Iwasaki & Simon 86a]. The significance of this approach is that it emphasises the way certain behaviour is achieved as opposed to simply verifying whether a certain parametric goal is achieved. In other words, the emphasis is on the dynamic behaviour of an artefact rather than static constraint satisfaction.

Such a behaviour-oriented view of design is the basic notion of our approach to design, and this is discussed more extensively in Chapter 5. However, we first define the language we shall use for this purpose in the following chapters. Readers not interested in the formal description of the temporal logic may proceed to Chapter 5 for discussion of behaviour based design.

2.7 Summary

In this chapter I have briefly introduced the background ideas concerning causal reasoning and some of the related technicalities. To summarise the points raised here.

- One aspect of causation is that it is knowledge about relations between events which is used in order to make it efficient when reasoning about events with temporal dimensions.
- When sharing an intuitive form of causal theory with causal rules representing *if cause then effect*, prediction can be done by deductive operations while abductive operations permit reasoning about the past.
- Causal reasoning involves various features of logics such as time, nonmonotonicity and possible worlds.
- Causal reasoning is applied in various fields where reasoning about causal relations and temporal sequence and ordering is important.

In the next chapter we examine Shoham's temporal logic; extend it to make it more expressive; and implement it so that it may be tested on sample problems.

Chapter 3

Reasoning about the future

To predict what will, or will not, follow the present situation is often central to efficient problem solving[1] and contributes towards a better management of causal knowledge. Reasoning about the future necessitates reasoning about events and their sequence. Since many philosophers believe, and it is widely conceived from everyday experiences, that time involves change [McTaggart 27, Shoemaker 69], it may be that prediction can be captured by reasoning about change.

Among the various approaches proposed to handle causal reasoning in computational terms, the temporal logic devised by Shoham [Shoham 88] is one of those computational and well defined languages. This chapter examines the efficacy of using temporal reasoning in the prediction task and introduces the approach taken by Shoham, extending his logic to enhance its expressiveness.

3.1 Prediction task

Prediction anticipates the future occurrence of events from information about the present. The agent only knows what is and isn't true up until the present time point and knows nothing about the future. If the agent knows or assumes the occurrence of an event in the future time point t_f and the task is to find out what would happen between the present and t_f, the problem is no longer prediction but interpolation (See

[1] It is not to claim that prediction itself is an efficient process.

section 4.2). In compliance with the formal notion of prediction provided in Section 2.3, a prediction task can be technically described as follows.

Let C be a set of events known to occur between a range of time points. If t_j is the *latest time point* (ltp)[2] among C, a *prediction task* derives a plausible occurrence of events at some t such that $t_j < t$, given an appropriate knowledge about the course of events.

This description of prediction, when inflicted upon the notion of past, present and future, preserves two important properties—'autonomy of the past' and 'defeasibility of future'. Prediction is based solely on the events in the past which cannot be altered by what has been predicted, and what is predicted is merely an assumption which can be reinstantiated when new information is added. Such characteristics of the prediction task in fact specify the mechanisms required to handle this task. The strict identification of temporal relations make it necessary to be provided with a facility to deal with the notion of time. To further complicate the matter, since certain conditions often change their values over time, it requires nonmonotonic reasoning. The complexity of these features is illustrated by the following problems.

Frame problem

This notorious problem pointed out by McCarthy[63], demonstrates the complexity of reasoning formalisms which deal with change. The problem is stated as follows [McCarthy & Hayes 69, p.487].

> If we had a number of actions to be performed in sequence we would have a number of conditions to write down that certain actions do not change the values of certain fluents [properties]. In fact with n actions and m fluents, we might have to write down mn such conditions.

This is essentially the problem of determining what *does not* change, without having to add an impractically large number of axioms which assert that certain conditions

[2]This notion follows the definition provided by Shoham[88] as reconstructed in Section 3.3.2.

do not alter the values of certain properties (*frame axioms*). One of the well known example is the Yale Shooting Problem (YSP) [Hanks & McDermott 86] described as follows[3]:

Example 1 (Yale shooting problem (YSP)) *Initially a gun is loaded. You wait for a while and fire the gun. The problem is whether it makes a noise.*[4]

Although the solution to this problem may seem simple, perhaps that there will indeed be a noise, there are many things to take into account when we assert this as the conclusion. What happens if the gun is unloaded sometime before it was fired? What if the firing event took place in a vacuum thus conveying no sound? These conditions can easily alter the course of events, and cannot be ignored altogether. From a different perspective, the frame problem is the problem of identifying what actually changes and what does not. According to Shoham, this problem can be further divided into the *qualification problem* and the *extended prediction problem*.

Qualification problem

This is the problem of identifying the almost infinite number of exceptions which can prevent an event from happening. Firing a gun in a vacuum is one such exception. However, listing all such conditions is computationally expensive—sometimes impossible. Devising a causal rule with an incredibly large antecedent and checking whether each condition in the antecedent is satisfied results in high complexity both in space and time. This makes it essentially the problem of a trade off between the amount of knowledge and the accuracy required in inference.

Extended prediction problem

This is the problem of formalising the prediction process over extended periods of time given a causal rule. For instance, if the gun is loaded, then we can predict that the gun

[3] Also called 'Hanks-McDermott problem'.

[4] There are variations in the result of shooting, from 'making a noise', 'killing a person' to 'killing a turkey.'

will be loaded in the following interval of time unless it is unloaded. But this rule must be applied over and over again in order to make any substantial prediction about the duration over which the gun is loaded. This process is again computationally complex, if not impractical.

An example related to this problem is Zeno's paradox[5] as pointed out by Rayner[89], though some, Rayner being one, may claim that this is a non problem but a confusion caused by failing to integrate a differential equation over an appropriate duration of time. Although this claim can be justified in a mathematical model, the extended prediction problem remains a problem in computational terms.

In the following section, we examine the idea put forward by Shoham to deal with the problems above.

3.2 Shoham's account of causal reasoning

Let us begin by giving a brief description of Shoham's logic and discussing the fundamental ideas proposed in the formalism. As described earlier, nonmonotonic logics are formulated by associating preference relations on models in a standard logic (propositional, first-order, etc.). By imposing a preference criterion, it is possible to specify, among many models, the class of preferred models. Shoham adopts this idea to solve the qualification problem.

The qualification problem occurs under the assumption that predictions are made possible only when enough details are given. In other words, we must specify every detail before we can predict anything useful. It is possible, however, to come up with a

[5]Strictly speaking, it refers to the Achilles paradox, one of the paradoxes by Zeno of Elea (ca 5th century B.C.). For those who are not familiar with this paradox: Suppose Achilles and a tortoise are having a race. Since it is obvious that Achilles is faster than the tortoise, the tortoise is starting at a distance ahead of Achilles. Suppose they started at the same time, it would take time t_1 for Achilles to reach the position where the tortoise started. In the meanwhile, the tortoise has travelled t_1 from that position, and is ahead of Achilles. Now it would require Achilles time t_2 to reach the position of the tortoise at time t_1, during which the tortoise travelled further for time t_2. To state this process in more general way, it would always take Achilles some finite amount of time to reach the position where tortoise is at that moment, and by the time Achilles reached there, the tortoise has travelled further. If we continue this process, Achilles will never catch the tortoise. It was not until the nineteenth century that this paradox was resolved by mathematicians.

prediction without exhaustive details, but only with the relevant information available. This prediction should be defeasible if new information is acquired.

The following discussion stems from this idea. Given a monotonic logic $\mathcal{L}$, there are numbers of models in the theory that describe the situation given in the Yale shooting problem. One such model is that there is a loud noise when the gun is fired; another is that the gun was unloaded manually before it was fired hence producing no noise. There may be a thousand other cases in which the gun makes no noise. But there is only one intended model, which is the one that makes a noise, without explicitly given such information as the gun being unloaded, the bullet replaced by something else, and so forth. The criterion of preference in this case should prefer the model in which "as little as possible happens". Such a monotonic logic $\mathcal{L}$ together with a preference criterion creates a nonmonotonic logic.

3.2.1 Chronological ignorance

The preference criterion as proposed by Shoham is that models in which *deflections*[6] occur as late as possible are preferred to the others. This criterion is supported by the following example. Given a loaded gun which is known to be fired at a later stage. The gun is loaded until it is fired. The intended model enables us to conclude that the gun makes a loud noise. In the second model, someone removes the trigger from the gun disabling the 'firing' event. In both cases, the number of deflections is one ('fire' for the first model and 'remove trigger' for the second), but the 'firing' event takes place *later* than the 'removing the trigger' event, thus preferring the first model. Figure 3.1 illustrates the temporal relation between these events.

Notice that we have assumed that this preference criterion is applied to chronologically minimise *all* propositions; we did not mention specifically which propositions to min imise. Shoham defines this generalised chronological minimisation to be *chronological ignorance*.

To give formal grounds to the notion of chronological ignorance, Shoham took two

[6] The divergence from the natural course of events.

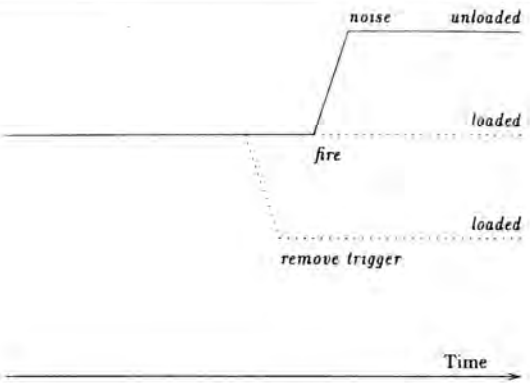


Figure 3.1: Chronological minimisation—the temporal relation between *fire* and *remove trigger* along with different scenarios for each case. The solid line indicates the intended model assuming *fire* event is known to happen.

steps. The first step was to construct a modal logic for temporal intervals, and the second step to incorporate the preference criterion and construct a nonmonotonic temporal logic.

Logic TK The logic constructed in the first step is the logic of temporal knowledge, or *logic TK*. It is essentially the modal logic version of a classical propositional interval logic (See Appendix A). Its semantics is based on the Kripke interpretation and all the possible worlds in the Kripke structure share the same interpretation of time as if they were parallel.

Logic CI By introducing the preference criterion of chronological minimisation into the logic TK, a model which is chronologically more minimal will be preferred over other models. As the result, the logic of chronological ignorance, or *logic CI* is constructed. The first-order case of the logics TK and CI are presented in Section 3.3.

3.2.2 Causal theories

When reasoning about causality, it is necessary to make use of knowledge about events and their relations. Such knowledge is generally described and in many cases formulated as *causal theories*, and provides a basis for knowledge about the natural course and occurrence of events.

Causal theories as defined by Shoham consist of two types of knowledge. First, there are causal rules, which each consist of antecedents and a consequent. Antecedents are further sorted as 'necessary' conditions and 'contingent' conditions, which are conjoined to form the left-hand side of the rule. The 'necessary' conditions are events which always need to be fulfilled to deduce the consequent, while 'contingent' conditions are events which are assumed to be true unless refuted. There is only one conclusion per rule and no conjunctions or disjunctions are allowed in the conclusion.

The second type of knowledge are boundary conditions, which are facts and the events known to happen. A boundary condition describes knowledge about the occurrence of an event or a fact which is not derived explicitly from the application of causal rules. Syntactically, it can be regarded as a causal rule without 'necessary' conditions, *i.e.*, an event that occurs unconditionally.

A wff in causal rules is made up of an operator, either 'necessary'($\Box$) or 'contingent'($\Diamond$), initial and terminal time points, and a proposition. The formal definitions of wff and the description of causal theories are given in the next section as a part of an extension to the first-order case.

Causal theories must be *sound* and *consistent*. By *sound* we mean that there are no two rules in a theory which have contradictory defaults, or, 'contingent conditions'. For instance, a causal theory containing both $\Box(t,t,a) \wedge \Diamond(t,t,p) \supset \Box(t+1,t+1,q)$ and $\Box(t,t,b) \wedge \Diamond(t,t,\neg p) \supset \Box(t+1,t+1,r)$ is not sound. By *consistent* we mean that there are no two rules in the same theory which have the same antecedent but contradictory consequents. The consistency condition guarantees that there is only one preferred model in logic CI for a causal theory ("The unique cmi theorem").

3.2.3 Potential histories

Shoham's notion of *potential histories* is based on that of *histories* introduced by Patrick Hayes[84]. According to this idea, a sequence of events can be represented by one or more *histories*. A history is associated with a *type* which is the period during which an object has the same qualitative physical behaviour.

The Yale Shooting Problem can be expressed in terms of histories as follows. In the beginning, there is a history that represents a loaded gun. After the gun is fired, a new history that the gun is unloaded begins. The overall behaviour of the gun is represented by the two sequential histories.

When it is applied to the prediction problem, Shoham points out two difficulties histories will suffer [Shoham 88]. First is the problem of synchronising the termination of two or more histories. In order to predict that a history persists until a new history takes over, there must be an axiom that guarantees this. If there are two histories that are supposed to end at the same time, an axiom that guarantees this become necessary. This is, however, not easy to formulate. The second problem is the extended prediction problem, which occurs in reasoning about a history's persistence or termination.

The basic notion of potential histories is the natural development of an event without any interference by other events. Unlike histories which are by all means true in a given circumstance, potential histories are valid as long as there is nothing that disturbs the natural course of persistence. When distracted, a potential history terminates and a new potential history should commence.

Chronological minimisation can be applied to the treatment of potential histories. The intended models are those that let potential histories last as long as possible without deflection. If a potential history is to be truncated, this should be done at the latest possible time point. In logic CI, this concept is incorporated into inertial rules which describe such knowledge. Inertial rules have the same structure as causal rules but the consequent is a wff expressing potential propositions. Such rules have the reading that a proposition persists between the initial and terminal time points, unless otherwise proved false. Projection rules interpret the potential rules to interpolate potential

propositions and their truth values. Potential boundary conditions are boundary conditions with potential histories.

Just as causal theories consisted of causal rules and boundary conditions, *inertial theories* are technically made up of inertial rules, projection rules and potential boundary conditions. However, I shall use the term 'causal theories' to refer to those made up of both causal and inertial theories, unless specifically stated.

3.3 Extension of Shoham's logic to the first-order case

Shoham did not go beyond the propositional case of the logic CI. In order to increase the expressiveness of the language, it is a natural progression to extend the logic to the first-order case, enabling representation of individuals and their properties. For practical applications, expressiveness is crucial in keeping the number of axioms to a manageable size.[7] In this section I describe the way to augment Shoham's propositional case to incorporate first-order features. A similar attempt was made independently by Bell[91]. To give it a stylistic consistency and to simplify comparison, the definitions and the presentation of theorems and proofs follow and resemble those of [Shoham 88].

3.3.1 The logic FOTK

Prior to the extension of the logic CI to the first-order case, we consider the logic FOTK, the first-order case for TK, and define its syntax and semantics. (See [Shoham 88, Section 4.2.1] for the original propositional case.)

The logic of temporal knowledge (the logic TK) is the monotonic version and its syntax and semantics are defined as described in Appendix A. Its syntax is the modal logic version of a classical propositional interval logic and the semantics is based on the Kripke interpretation of the formulae; in this case, every possible world in the Kripke structure shares the same interpretation of time, as if they were parallel worlds. Here, the truth value assignments in the same interval of time may be different in distinct

[7] The only example Shoham actually provided with causal theory was the Yale shooting problem, which consisted of a mere 5 axioms.

worlds.

Syntax

Given:

TC: a set of time point symbols, namely integers,

C: a set of constant symbols that is disjoint from TC,

TV: a set of temporal variables,

V: a set of variables that is disjoint from TV,

TF: a set of temporal function symbols (typical ones being the arithmetic operators), each associated with a fixed arity

F: a set of function symbols that is disjoint from TF, each associated with a fixed arity, and

R: a set of relation symbols, each with a fixed arity.

Note that the interpretation set for time is always defined to be the set of integers.

The set of *temporal terms* is defined inductively as follows:

1. All members of TC are temporal terms.
2. All members of TV are temporal terms.
3. If $trm_1, ..., trm_n$ are temporal terms, and $f \in TF$ is an n-ary function symbol, then $f(trm_1, ..., trm_n)$ is a temporal term.

The set of *non temporal terms* is defined in exactly the same way, with TC replaced by C, TV replaced by V, and TF replaced by F.

The set of well formed formulas (wffs) is defined inductively as follows:

1. If trm_a and trm_b are temporal terms, then $trm_a = trm_b$ and $trm_a \leq trm_b$ are wffs.

2. If trm_a and trm_b are temporal terms, $trm_1, ..., trm_n$ are non-temporal terms, and $r \in R$ an n-ary relation symbol, then $\text{TRUE}(trm_a, trm_b, r(trm_1, ..., trm_n))$ is a wff.

3. If φ_1 and φ_2 are wffs then so are $\varphi_1 \wedge \varphi_2$, $\neg\varphi_1$ and $\Box\varphi_1$.

4. If φ is a wff and $z \in TV \cup V$ is a variable, then $\forall z\varphi$ is a wff.

The usual definitions are assumed for $\vee, \supset, \equiv, \exists$, etc. Also, $\Diamond$ is defined by $\Diamond\varphi \equiv \neg\Box\neg\varphi$.

Semantics

As in the propositional case, in TK, a *Kripke interpretation* is a set of infinite worlds, where the same interpretation of time is shared. Different worlds have different variable assignments, and truth values.

First let us begin by the reminder of the definition of sets (as in [Shoham 88] for the non-modal case).

- TU is a nonempty universe of time points,
- $\leq$ is a binary relation on TU,
- U is a nonempty universe of individuals that is disjoint from TU,
- TFN is a set of total functions in $\bigcup_k (TU^k \rightarrow TU)$,
- FN is a set of total functions in $\bigcup_k (U^k \rightarrow U)$,
- RL is a set of relations over U

A *Kripke interpretation* is a pair (W, M), where W is a nonempty universe of *possible worlds*, each consisting of the individuals (U), non-temporal functions (FN), and non-temporal relations (RL). $\mathcal{N}$ is a structure consisting of the time points TU which we

take to be the integers ($\mathcal{N} \rightarrow TU$), the temporal relation $\leq$ (which we take as the natural order on the integers), and temporal functions (TFN).

$M = (M1, M2, M3, M4, M5)$ is a meaning function as follows:

$M1 : TC \rightarrow TU$,
$M2 : C \rightarrow U$,
$M3 : TF \rightarrow TFN$,
$M4 : TU \times TU \times F \rightarrow FN$,
$M5 : w \times TU \times TU \times R \rightarrow RL$.

In the modal case, a *variable assignment* is a function $VA = (VAT, VAV)$, such that $VAT : TV \rightarrow \mathcal{N}$ and $VAV : V \rightarrow U$.

M and VA induce a time-dependent meaning MVA on arbitrary terms in the following way. An *arbitrary temporal term* is treated so that its meaning is the same regardless of when the terms are interpreted. If $tv \in TV$ then
$MVA(tv) = VAT(tv)$. If $tc \in TC$ then $MVA(tc) = M1(tc)$.
If $f \in TF$ and $trm = f(trm_1, ..., trm_n)$ is a temporal term, then
$MVA(trm) = M3(f)\ (\ MVA(trm_1), ..., MVA(trm_n)\ )$.

An *arbitrary non-temporal term* is treated so that meaning depends on the time of interpretation. If $v \in V$, then for all $trm_a, trm_b \in TU$, $MVA(trm_a, trm_b, v) = VAV(v)$. If $c \in C$ then $MVA(trm_a, trm_b, c) = M2(c)$. The temporal dependence of the interpretation is incorporated in the following manner: if $f \in F$ and $trm = f(trm_1, ..., trm_n)$ is a non-temporal term, then $MVA(trm_a, trm_b, trm) =$
$M4(trm_a, trm_b, f)(MVA(trm_a, trm_b, trm_1), ..., MVA(trm_a, trm_b, trm_n))$.

A Kripke interpretation $KI = (W, M)$ and a world $w \in W$ satisfy a wff φ under the variable assignment VA (written $KI, w \models \varphi[VA]$) under the following inductively defined conditions.

- $KI, w \models trm_1 = trm_2[VA]$ iff $MVA(trm_1) = MVA(trm_2)$.
- $KI, w \models trm_1 \leq trm_2[VA]$ iff $MVA(trm_1) \leq MVA(trm_2)$.

- $KI, w \models \text{TRUE}(trm_a, trm_b, r(trm_1, ..., trm_n))[VA]$ iff $\langle MVA(trm_1), ..., MVA(trm_n)\rangle \in M5(w, MVA(trm_a), MVA(trm_b), r)$.
- $KI, w \models (\varphi_1 \wedge \varphi_2)[VA]$ iff $KI, w \models \varphi_1[VA]$ and $KI, w \models \varphi_2[VA]$.
- $KI, w \models (\neg\varphi)[VA]$ iff $KI, w \not\models \varphi[VA]$.
- $KI, w \models (\forall z\varphi)[VA]$ iff $KI, w \models \varphi[VA']$ for all VA' that agrees with VA everywhere except possibly on z.
- $KI, w \models \Box\varphi[VA]$ iff $KI, w' \models \varphi[VA]$ for all $w' \in W$.

Again, as in the propositional case, from the identity of time and other universally quantified variables across worlds (assuming $S5$) follows the validity of the *Barcan formula*, *i.e.*, $\Box\forall v\varphi \equiv \forall v\Box\varphi$

3.3.2 The logic FOCI

To incorporate a preference criterion to FOTK, nonmonotonic feature becomes necessary to the logic. This nonmonotonic version is called the logic FOCI after Shoham's logic of chronological ignorance (CI). For simplicity, however, while retaining a first-order feature we restrict ourselves to the case where all nontemporal variables are instantiated; i.e., we focus on the case where there are no nontemporal variables. In FOCI, the notion of the *latest time point* (or *ltp*) is introduced to play an important role in the preference criterion. For a base formula, where all nontemporal variables are instantiated, the ltp is defined as "the chronologically latest point mentioned in it" (see Definition 4 for the formal definition). The preferred model, a chronologically more ignorant model in this case, is then defined to be the model where less is known about later events. The model preferred most is then called the *chronologically maximally ignorant* model (the *cmi* model).

As in the propositional case, the logic FOTK is modified into a nonmonotonic version, the logic FOCI. First, the definitions of base wffs, base ground wffs, and the latest time point (ltp) are given. (See [Shoham 88, Section 4.2.2] for the original propositional case.)

Definition 1 (Base wffs) Base wffs *are those containing no occurrence of the modal operator. [Shoham 88, p.105].*

Definition 2 (Base ground wffs) Base ground wffs *are base wffs with no uninstantiated variables.*

Definition 3 (Latest time points) *The* latest time point (ltp) *of a base formula is the latest time point mentioned in it, formally defined as follows. Let eval(trm) stand for "the evaluated value of trm", which is mapped to an integer representing a time-point,*

1. *The* ltp *of* $\text{TRUE}(trm_a, trm_b, r(trm_1, ..., trm_n))$ *(where* trm_a *and* trm_b *are temporal terms,* $trm_1, ..., trm_n$ *are non-temporal terms, and* r *is a relation symbol) is* $max\{eval(trm_a), eval(trm_b)\}$.
2. *The* ltp *of* $\varphi_1 \wedge \varphi_2$ *is the latest (w.r.t. the standard interpretation of time) between the* ltp *of* φ_1 *and the* ltp *of* φ_2.
3. *The* ltp *of* $\neg\varphi$ *is the* ltp *of* φ.
4. *The* ltp *of* $\forall v\varphi$ *is the earliest among the* ltps *of all* φ' *which result from substituting and evaluating in* φ *a temporal term for all free occurrences of* v, *or* $-\infty$ *if there is no such earliest* ltp.

Because we restrict ourselves to the case where there are no nontemporal variables, we can use Shoham's definitions of "chronologically more ignorant" and "chronologically maximally ignorant" models [Shoham 88, p.106].

Definition 4 (Chronologically more ignorant) *A Kripke interpretation* M_2 *is* chronologically more ignorant *than a Kripke interpretation* M_1 *(written* $M_1 \sqsubset_{ci} M_2$*) if there exists a time* t_0 *such that*

1. *for any base sentence* φ *whose* ltp $\leq t_0$, *if* $M_2 \models \Box\varphi$ *then also* $M_1 \models \Box\varphi$, *and*

2. *there exists some base sentence* φ *whose* ltp *is* t_0 *such that* $M_1 \models \Box\varphi$ *but* $M_2 \not\models \Box\varphi$.

Definition 5 (Chronologically maximally ignorant) *M is said to be a* chronologically maximally ignorant *(or* cmi*)* model *of φ if $M \models_{\sqsubset_{ci}} \varphi$, that is, if $M \models \varphi$ and there is no other M' such that $M' \models \varphi$ and $M \sqsubset_{ci} M'$.*

As in the propositional case, the logic FOCI is the nonmonotonic extension to the logic FOTK. It shares the same syntax as FOTK, but places preference to chronologically more ignorant models.

3.4 Causal theory in FOCI

Based on the definitions of the logic CI for the first-order case, the causal theory can be defined as below.

Definition 6 (Causal theory) *A causal theory Ψ is a theory in CI, in which all formulae have the form*

$$\Phi \wedge \Theta \supset \Box\varphi$$

where

- *φ is a (positive or negative) atomic base formula* $\text{TRUE}(trm_a, trm_b, p^k)$[8] *(and by convention, $eval(trm_a) \leq eval(trm_b)$).*
- *Φ is a (possibly empty) conjunction of expressions $\Box\varphi_i$, where φ_i is a (positive or negative) atomic base formula. These expressions provide 'necessary' conditions of the formula.*
- *Θ is a (possibly empty) conjunction of expressions $\Diamond\varphi_i$, where φ_i is a (positive or negative) atomic base formula. These expressions provide 'contingent' conditions of the formula.*

[8] p^k denotes a k-ary predicate.

Definition 7 (Causal rules, boundary conditions) Boundary conditions *are formulae in a causal theory Ψ with empty Φ. All other formulae (i.e. with non empty Φ are called* causal rules.

Note:

1. The symbol $[\neg]$ is an optional appearance of the negation sign.
2. $\Box(trm_a, trm_b, p^k)$ and $\Box(trm_a, trm_b, \neg p^k)$ are equivalent to $\Box\text{TRUE}(trm_a, trm_b, p^k)$ and $\Box\neg\text{TRUE}(trm_a, trm_b, p^k)$ respectively. The same applies for the $\Diamond$ operator.
3. Base formulae in CI are those with no modal operators and refer directly to the real world and not to knowledge of the world. Atomic base formulae are the base formulae of the forms $\text{TRUE}(trm_a, trm_b, p^k)$ (*positive*) or $\text{TRUE}(trm_a, trm_b, \neg p^k)$ (*negative*).
4. The *ltp* of a base formula is the latest time mentioned or evaluated in it, or, if the formula contains a quantifier, the earliest among the *ltp*s of all sentences resulting from replacing the quantified variable by a constant.

Definition 8 (Soundness) *A causal theory Ψ is* sound *if there do not appear in any antecedents of two formulae in Ψ, $\Diamond(trm_{1a}, trm_{1b}, P_1)$ and $\Diamond(trm_{2a}, trm_{2b}, P_2)$, where $eval(trm_{1a}) = eval(trm_{2a})$, $eval(trm_{1b}) = eval(trm_{2b})$, and predicates P_1 and P_2 are contradictory.*

Note that since $\Diamond$-conditions represent defaults in a causal theory, the soundness condition ensures that there are no contradictory defaults within a causal theory.

Definition 9 (Consistency) *A causal theory Ψ is* consistent *if there do not exist two formulae in Ψ, whose antecedents are the same and the consequents are contradictory. For example, $\Phi \wedge \Theta \supset \Box(trm_{1a}, trm_{1b}, P_1)$ and $\Phi \wedge \Theta \supset \Box(trm_{2a}, trm_{2b}, P_2)$ where $eval(trm_{1a}) = eval(trm_{2a})$, $eval(trm_{1b}) = eval(trm_{2b})$, and predicates P_1 and P_2 are contradictory, are inconsistent.*

3.4.1 The unique cmi model

According to the definition of the causal theory stated above, the "unique cmi model" theorem can be stated as follows. [Shoham 88, p.112].

Theorem 1 (The "unique cmi model" theorem) *If Ψ is any causal theory, then*

1. *Ψ has a cmi model, and*
2. *if M_1 and M_2 are both cmi models of Ψ, and φ any base sentence, then $M_1 \models \Box\varphi$ iff $M_2 \models \Box\varphi$.*

Proof: As in the propositional case, the proof is by the construction of a model M for Ψ such that M is chronologically more ignorant than any model for Ψ which differs from M on the truth value of some sentence $\Box\varphi$, where φ is a base sentence.

The construction starts with some time bounded Kripke interpretation M/t_0, where M/t stands for a model M which holds up until time t, and augments it to later time points iteratively:

1. Let t_0 be a time point preceding the *ltp* of any sentence φ such that $\Theta \supset \Box\varphi$ is a boundary condition. For any base tautology φ whose $ltp \leq t_0$, let $M/t_0 \models \Box\varphi$. For any other base wff φ whose $ltp \leq t_0$, let $M/t_0 \not\models \Box\varphi$.
   Notice that M/t_0 (partially) satisfies the boundary conditions of Ψ vacuously and (partially) satisfies all the causal rules.

2. We now specify how to augment $M/t+1$ for any t. Let
   $\text{CONSEQUENTS}_{t+1} = \{\ \Box(trm_a, trm_b, r(trm_1, ..., trm_n))$:
   $\Phi \wedge \Theta \supset \Box(trm_a, trm_b, r(trm_1, ..., trm_n)) \in \Psi, eval(trm_b) = t+1,$
   $M/t \models \Phi \wedge \Theta\ \}$.

   $M/t+1$ is obtained by making all wffs in CONSEQUENTS_{t+1} and all their tautological consequents true, and for any other base wff φ' whose *ltp* is evaluated to be $t+1$, making $\Box\varphi'$ false.

Clearly, $M/\infty\,(M)$ is a model for Ψ. To conclude the proof, it remains to show that if a model M' differs from M on the truth value of $\Box\varphi$ for some base sentence φ, then M' is not a cmi model. To prove this, suppose that some model M' of Ψ indeed differs from M on the truth value of $\Box\varphi$ for some base sentence φ.

1. If $M' \models \Box\varphi$ for some nontautological base sentence φ whose $ltp \leq t_0$, then M' is chronologically less ignorant than M.

2. Otherwise, there is an earliest t such that $t_0 \leq t$, and such that for some base sentence φ whose ltp is $t+1$, M and M' differ on the truth value of $\Box\varphi$. There are two possible cases:

   (a) $M \models \Box\varphi$ and $M' \not\models \Box\varphi$. By the construction procedure, if $M \models \Box\varphi$ then there exists a sentence $\Phi \wedge \Theta \supset \Box\varphi \in \Psi$, such that the ltps of the base sentences in Φ and Θ are $\leq t$, and such that $M \models \Phi \wedge \Theta$. Since M and M' agree on the knowledge of all base sentences whose $ltp \leq t$, it must be the case that also $M' \models \Phi \wedge \Theta$. But since $M' \not\models \Box\varphi$, M' cannot be a model for Ψ; contradiction.

   (b) $M' \models \Box\varphi$ and $M \not\models \Box\varphi$. From the first case it is known that for any φ' whose ltp is $t+1$, if $M \models \Box\varphi'$ then also $M' \models \Box\varphi'$. But these two facts together imply that M' is chronologically less ignorant than M.

   □

It is important to note here the following issue in Shoham's formulation. According to the definition of ltp (p.36), $-\infty$ is considered as a possible value. This causes a serious problem in the step 1 of the proof above since there is a possible case of $-\infty \leq t_0$ which leads to the consideration of $M/-\infty$. Also, while the inductive step (step 2) describes the construction of $M/t+1$ from M/t, by nature of inductive proof, this is not adequate to show that $M/\infty\,(M)$ is well defined. Practically, however, these problems do not cause difficulties since any number used to identify a time step is always finite.

To analyse the complexity of a causal theory, let us first define a *finite* causal theory.

Definition 10 (Finite causal theory) *In a* finite causal theory,

- *there is no occurrence of function symbols,*
- *the duration of the model is bounded between time points t_1 and t_2 so that any instantiation of temporal variables should lie between t_1 and t_2, and*
- *there is no occurrence of non-temporal variables.*

Since function symbols are disallowed the Herbrand Base will be of finite size.

Theorem 2 (Complexity of a finite causal theory) *Let Ψ be a* finite *causal theory of size n.*[9] *The (unique) set of base sentences that are true in any cmi model of Ψ is of size $O(n)$, and can be computed in time $O(n \log n)$.*

Proof: Intuitively, the following algorithm organises all sentences (axioms) and atomic base sentences of Ψ in ascending order of *ltp*, checking the axioms in order determining whether the atomic base sentences can be proved to be true (or false). This is possible since we restrict all the variables to be grounded. This procedure can be regarded the same as the propositional case, since the first-order predicates appearing here are ground and can be treated propositionally. Hence, the basic algorithm of the proof emulates the propositional case. The details of the algorithm are shown below.

1. Let T be the list of all axioms in Ψ. Gather all atomic base sentence appearing in T into a list S, dropping negation signs and removing duplicates. Furthermore, assign all possible individual constants to every free variable occurring inside the non-temporal terms (*viz.* the predicates). In other words, if
   $\Psi \wedge \Theta \supset \Box(trm_a, trm_b, [\neg]r(trm_1, ..., trm_n)) \in T$ and
   $eval(trm_a) = t_1, eval(trm_b) = t_2$, then $\text{TRUE}(t_1, t_2, r(trm'_1, ..., trm'_n))$
   for every possible assignment of $trm'_i (1 \leq i \leq n)$.

[9] Practically, the *size* of a causal theory is determined as the number of distinct (instantiated) base sentences that may be formed by instantiating axioms. Thus it is dependent not only on the number of axioms but on the whole time span of inference, the number of elements in the set of individuals.

2. Sort T and S by *ltp*. Label all members of S UNMARKED.

3. If T is empty then halt. The (positive and negative) variable-free base sentences that are known in the cmi models of Ψ are exactly those marked *YES* in S, and the negation of those marked *NO* in S. Otherwise,

4. Remove the first element of T, say $\Phi \wedge \Theta \supset \Box(trm_a, trm_b, [\neg]r(trm_1, ..., trm_n))$. For each conjunct $\Box(trm_{i_a}, trm_{i_b}, [\neg]r_i(trm_{i_1}, ..., trm_{i_n}))$ of Φ and each conjunct $\Diamond(trm_{i_a}, trm_{i_b}, [\neg]r_i(trm_{i_1}, ..., trm_{i_n}))$ of Θ, determining how $\text{TRUE}(trm'_{i_a}, trm'_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ ($eval(trm'_{i_a}) = trm_{i_a}$ and $eval(trm'_{i_b}) = trm_{i_b}$) is marked in S, performing binary search on S. If one of the following conditions is true:

   (a) $\Box(trm_{i_a}, trm_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ is a conjunct of Φ and $\text{TRUE}(trm_{i_a}, trm_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ is not marked *YES* in S

   (b) $\Box(trm_{i_a}, trm_{i_b}, \neg r_i(trm_{i_1}, ..., trm_{i_n}))$ is a conjunct of Φ and $\text{TRUE}(trm_{i_a}, trm_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ is not marked *NO* in S

   (c) $\Diamond(trm_{i_a}, trm_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ is a conjunct of Θ and $\text{TRUE}(trm_{i_a}, trm_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ is marked *NO* in S

   (d) $\Diamond(trm_{i_a}, trm_{i_b}, \neg r_i(trm_{i_1}, ..., trm_{i_n}))$ is a conjunct of Θ and $\text{TRUE}(trm_{i_a}, trm_{i_b}, r_i(trm_{i_1}, ..., trm_{i_n}))$ is marked *YES* in S

   then go to 3.

   Otherwise, label $\text{TRUE}(trm_a, trm_b, r(trm_1, ..., trm_n))$ in S with *YES* or *NO*, depending on whether the consequent is $\Box(trm_a, trm_b, r(trm_1, ..., trm_n))$ or $\Box(trm_a, trm_b, \neg r(trm_1, ..., trm_n))$.

   Go to 3.

Complexity analysis: Since T is finite, the algorithm clearly must halt. Its time complexity is identical to the propositional case with the overall time complexity being $O(n \log n)$.

Step1: $O(n)$.

Step2: $O(n \log n)$ (sorting).

Step3: $O(1)$.

Step4: Checking the existing labelling is done at most n times during the entire execution of the algorithm, and every individual checking is done in time $O(n \log n)$ (binary search). Similarly, at most n new labels are made throughout the algorithm, and again each labelling can be done in time $O(\log n)$. Total complexity: $O(n \log n)$.

3.5 Inertial theory in FOCI

The definition of inertial theory for the first order case can be stated as below. Essentially, an inertial theory is created on top of causal theory (Ψ_1) replacing the frame axioms with potential histories (Ψ_2) and the rules that interpolate the events from potential histories (Ψ_3), as in the propositional case.

Definition 11 (Inertial theory) *An inertial theory is a collection of formulae* $\Psi_1 \cup \Psi_2 \cup \Psi_3$ *such that:*

I. Ψ_1 *is a causal theory,*

II. Ψ_2 *is a collection of formulae of the form*

$\Phi \wedge \Theta \supset \text{POTEN}(trm_a, trm_b, \text{p-}i^k) =_{def}$

$\Phi \wedge \Theta \supset \exists v(eval(trm_a) \leq v \leq eval(trm_b) \wedge \Box(trm_a, v, \text{p-}i^k))$

such that

1. $\text{p-}i^k$ *is a k-ary first-order formula denoting a potential history*
2. *If* $\Phi \wedge \Theta \supset \text{POTEN}(trm_a, trm_b, \text{p-}i^k)$ *is in* Ψ_2, *then* $\text{p-}i^k$ *may appear only on the r.h.s. of formulae in* Ψ_2 *(or in* Ψ_3 *as explained below).*
3. Φ *is a conjunction of formulae* $\Box\varphi_i$, *where* φ_i *is a (positive or negative) atomic base formula with* $\text{ltp}\ t_i$ *such that* $t_i \leq eval(trm_a)$.

4. Θ *is a conjunction of formulae* $\Diamond\varphi_j$, *where* φ_j *is a (positive or negative) atomic base formula with* ltp t_j *such that* $t_j \leq eval(trm_a)$.

5. *Either or both of* Φ *and* Θ *may be empty (i.e., identically true). A formula in which* Φ *is empty is called an* inertial boundary condition. *Other formulae are called* inertial causal rules.

6. *There is a time point* t_0 *such that if* $\Theta \supset$ POTEN$(trm_a, trm_b, \mathrm{p}{-}i^k)$ *is an inertial boundary condition, then* $t_0 \leq eval(trm_a)$.

7. *There do not exist two formulae in* $\Psi_1 \cup \Psi_2$ *such that one contains* $\forall z \Diamond \varphi$ *on its l.h.s. and the other contains* $\forall z \Diamond \neg\varphi$ *on its l.h.s. (*Soundness *condition).*

III. Ψ_3 *is a collection of formulae of the form*

PROJECT$(trm_a, \mathrm{p}{-}i^k, trm_b, trm_c, [\neg]p^k) =_{def}$

$(\exists v(trm_c \leq v \wedge \Box(trm_a, v, \mathrm{p}{-}i^k))) \supset \Box(trm_b, trm_c, [\neg]p^k)$

such that

1. $eval(trm_a) \leq eval(trm_b) \leq eval(trm_c)$

2. *If* $X \supset$ POTEN$(trm_a, trm_d, \mathrm{p}{-}i^k)$ *is a formula in* Ψ_2,
   PROJECT$(trm_a, \mathrm{p}{-}i^k, trm_a, trm_a, p^k)$ *is in* Ψ_3, *and*
   $Y \supset \Box(trm_a, trm_a, \neg p^k)$ *is in* Ψ_1, *then* $X \wedge Y$ *is inconsistent.*

3. *If* $X_1 \supset$ POTEN$(trm_a, trm_d, \mathrm{p}{-}1^k)$ *and* $X_2 \supset$ POTEN$(trm_e, trm_f, \mathrm{p}{-}2^k)$ *are formulae in* Ψ_2, PROJECT$(trm_a, \mathrm{p}{-}1^k, trm_b, trm_c, p^k)$ *and*
   PROJECT$(trm_e, \mathrm{p}{-}2^k, trm_g, trm_h, \neg p^k)$ *are formulae in* Ψ_3 *such that* $eval(trm_a) \leq eval(trm_e)$, *and* $X_1 \wedge X_2$ *is consistent, then the following must hold:*

   (a) *If* $eval(trm_e) = eval(trm_b) = eval(trm_c)$ *then* $eval(trm_a) \leq eval(trm_e)$

   (b) *Otherwise, for some* $eval(trm_h) \leq eval(trm_c)$ *and some* q,
   PROJECT$(trm_e, \mathrm{p}{-}i^k, trm_g, trm_h, q^k)$ *is in* Ψ_3 *and*
   $X_j \supset \Box(trm_g, trm_h, \neg q^k)$ *is in* Ψ_1, *for* $i = 1, j = 2$ *or vice versa.*

3.5.1 The unique cmi model

As in the case for the causal theory, the inertial theory in the first-order case has a "unique" cmi model as can be proved as follows (after [Shoham 88, p.136]).

Theorem 3 (The "unique cmi model") *If $\Psi = \Psi_1 \cup \Psi_2 \cup \Psi_3$ is any inertial theory above , then*

1. *Ψ has a cmi model, and*
2. *if M_1 and M_2 are both cmi models of Ψ, and φ any base formula, then $M_1 \models \Box\varphi$ iff $M_2 \models \Box\varphi$.*

Proof: As in the propositional case, the proof is by construction. The sets manipulated continue to have the same properties as before and the descriptions here are limited to the definitions of the augmentations of each set from time t to $t+1$ (temporal terms evaluated).

NEW-POTENTIALS$_{t+1}$ =
$\{\langle t+1, t_1, \mathrm{p}{-}i^k\rangle :$
$\Phi \wedge \Theta \supset$ POTEN$(trm_a, trm_b, \mathrm{p}{-}i^k) \in \Psi_2$ and
$M/t \models \Phi \wedge \Theta$.
$eval(trm_a) = t+1$ and $eval(trm_b) = t_1\}$

PARTIAL-CONSEQUENTS$_{t+1}$ =
$\{\Box\varphi$:
the *ltp* of φ is $t+1$,
$\Phi \wedge \Theta \supset \Box\varphi \in \Psi_1$, and
$M/t \models \Phi \wedge \Theta\}$
$\cup$
$\{\Box(t+1, [\neg]p^k)$:
$\langle t+1, t_1, \mathrm{p}{-}i^k\rangle \in$ NEW-POTENTIALS$_{t+1}$, and

PROJECT(trm_a, p$-i^k$, trm_b, trm_c, $[\neg]p^k$) $\in \Psi_3$,
where $eval(trm_a) = eval(trm_b) = eval(trm_c) = t + 1$ }

NATURAL-DEATH$_t$ =
{$\Box(t_1, t$, p$-i^k$):
$\langle t_1, t$, p$-i^k\rangle \in$ POTENTIALS$_t$ }

CLIPPED$_t$ =
{$\Box(t_1, t$, p$-i^k$):
$\langle t_1, t_2$, p$-i^k\rangle \in$ POTENTIALS$_t$,
PROJECT(trm_a, p$-i^k$, trm_b, trm_c, $[\neg]p^k$) $\in \Psi_3$
where $eval(trm_a) = t_1, eval(trm_b) = t_3, eval(trm_c) = t + 1$, and
$\Box(t_3, t + 1, \neg[\neg]p^k) \in$ PARTIAL-CONSEQUENTS$_{t+1}$ }

POTENTIALS$_{t+1}$ =
(NEW-POTENTIALS$_{t+1} \cup$ POTENTIALS$_t$)$-$
{$\langle t_1, t_2$, p$-i^k\rangle$:
$\Box(t_1, t$, p$-i^k) \in$ (NATURAL-DEATH$_t \cup$ CLIPPED$_t$) }

CONSEQUENTS$_{t+1}$ =
PARTIAL-CONSEQUENTS$_{t+1}\cup$
{$\Box(t_3, t + 1, [\neg]p^k)$:
$\langle t_1, t_2$, p$-i^k\rangle \in$ POTENTIALS$_{t+1}$, and
PROJECT(trm_a, p$-i^k$, trm_b, trm_c, $[\neg]p^k$) $\in \Psi_3$,
where $eval(trm_a) = t_1, eval(trm_b) = t_3, eval(trm_c) = t + 1$ }

The set CONSEQUENTS are used in the same manner as were in the propositional case to create the cmi model. The remaining part of the proof that "if a model M' differs from M on the truth value of $\Box\varphi$ for some base sentence φ, then M' is chronologically

less ignorant than M" is omitted here since there are no significant difference from the propositional case.

Complexity analysis: The complexity of the computation is again $O(n \log n)$ for a finite set of axioms for inertial theory with size n. As was in the causal theory, the function symbols are not allowed in order to guarantee the restriction to be *finite*.

The basic procedures are sorting and searching, and both can be computed in $O(n \log n)$.

3.6 Implementation

The extended language of CI is implemented in SICStus Prolog to test its capability and computational characteristics. The algorithm to compute the cmi models essentially follows the proof of the unique cmi theorem. In order to achieve the complexity as desired in the complexity analysis, efficient algorithms such as merge sorting and binary search algorithm [Aho *et al* 83] were applied. The actual implementation first checks for the well-formedness of causal theories and the conditions they must satisfy. These are soundness and consistency conditions, which must be checked for the whole theory. In order to guarantee termination, no function symbol is introduced and the temporal terms map to the set of discrete time-point symbols, in this case, integers.

Both propositional and first-order cases were implemented, basically in a similar manner. The only difference between the algorithm for the two cases is the creation of the set of atomic base sentences (set S) and the set of axioms (set T). In the propositional case, only the temporal terms were universally quantified, and the quantifiers were removed by collecting the sentences for possible time-points. On the other hand, in the first-order case, the non-temporal variables are quantified in addition to the temporal variables, and the collection of the members of sets S and T involves not only the expansion over duration of the inference but also over the individuals. These expansion processes are in fact computationally expensive. The assignment of temporal terms, for instance, involves the following procedures.

Since the algorithm for the causal theory requires that all *atomic* base sentences and

instantiated axioms be collected in sets (S and T, respectively), the universal quantifiers over time must be removed by actually assigning the time points over the time span of the inference. For instance, if the duration of inference is between the interval of $t = 1$ to $t = 3$, then an axiom

$$\Box(t,t,p) \wedge \Diamond(t,t,q) \supset \Box(t+1,t+1,r)$$

must be collected in set T as

$$\Box(1,1,p) \wedge \Diamond(1,1,q) \supset \Box(2,2,r)$$

$$\Box(2,2,p) \wedge \Diamond(2,2,q) \supset \Box(3,3,r)$$

$$\Box(3,3,p) \wedge \Diamond(3,3,q) \supset \Box(4,4,r)$$

and corresponding set S should have as its members all nine atomic base sentences, *i.e.*, TRUE$(1,1,p)$, TRUE$(1,1,q)$, TRUE$(2,2,p)$, TRUE$(2,2,q)$, TRUE$(2,2,r)$, TRUE$(3,3,p)$, TRUE$(3,3,q)$, TRUE$(3,3,r)$, TRUE$(4,4,r)$.

Obviously, as the number of axioms becomes larger, and the duration of inference longer, the sizes of the sets grow exponentially. And the construction of these sets is no less computationally expensive than sorting or search. The assignment of individuals in the first-order case increases this complexity. However, the implementation of inertial theories does not suffer from the complexity of temporal assignment since there is no need to construct sets T and S and the assignment relies on the built-in unification process of Prolog. Actual Prolog code to compute the cmi models for causal theories which include inertial rules can be found in the AppendixC.1.

3.7 Sample problems for extended CI

In this section, I demonstrate the formulation of qualitative reasoning problems in the framework of extended CI (the results described here are partly reported in [Nakata 94b]).

3.7.1 Simple oscillator

One of the features of *qualitative reasoning* is its capability to predict[10] the behaviour of a system. The problems in the domain of qualitative reasoning involving such a task, therefore, should be able to be translated into the language of the causal theory.

A behaviour analysis of an oscillator, a block and a spring, is one of the well known problems in qualitative reasoning [Forbus 84]. The behaviour is described as a transition in a state-space, each state featured with acceleration and velocity, and their qualitative values. By following the transition, it is possible to predict the behaviour of the system.

Scenario The scenario of the *simple oscillator problem* is:

> *There is a block which is placed on a surface with no friction. The block is connected to a wall with a spring. Initially, the block is held at the place where the spring is stretched. Once it is released, the block should move between the points where the spring is stretched and compressed, unless the spring breaks.*

Figure 3.2 describes the block; for convenience, the position of the block is identified as in the figure.

Formulation The problem is formulated without using potential histories for simplicity, making clear the frame axioms. The facts and events are expressed propositionally. The formulation begins with identifying the boundary conditions, then causal rules, and finally persistence conditions.

Boundary conditions

Initially, the block is held at the position "stretched" and its force, velocity and acceleration are all zero.

[10] Strictly following the terminology used in the field, *envisioning* the behaviour of a system.

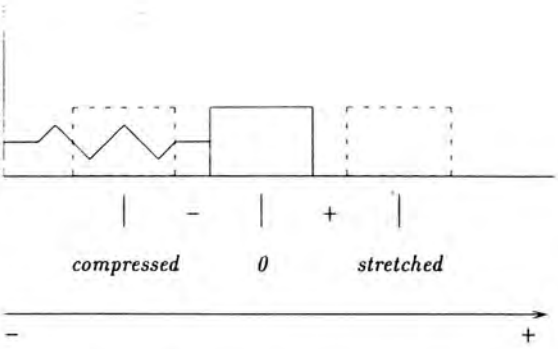


Figure 3.2: A simple block oscillator.

1. □(1,1,stretched)
2. □(1,1,v=0)
3. □(1,1,a=0)
4. □(1,1,f=0)
5. □(1,1,held)

The block is released at time-point 3.

6. □(3,3,released)

Persistence conditions

The block is held unless released

7. □(t,t,held) ∧ ◇(t,t,¬released) ⊃ □(t+1,t+1,held)

While the block is held, the force is balanced to be zero:

8. $\Box(t,t,held) \wedge \Diamond(t,t,\neg released) \wedge \Diamond(t,t,\neg springbreak) \supset \Box(t+1,t+1,f=0)$

No change in position unless the block has non-zero velocity:

9. $\Box(t,t,stretched) \wedge \Diamond(t,t,\neg v=-) \supset \Box(t+1,t+1,stretched)$

10. $\Box(t,t,p=+) \wedge \Diamond(t,t,\neg v=-) \wedge \Diamond(t,t,\neg v=+) \supset \Box(t+1.t+1,p=+)$

11. $\Box(t,t,p=0) \wedge \Diamond(t,t,\neg v=-) \wedge \Diamond(t,t,\neg v=+) \supset \Box(t+1,t+1,p=0)$

12. $\Box(t,t,p=-) \wedge \Diamond(t,t,\neg v=-) \wedge \Diamond(t,t,\neg v=+) \supset \Box(t+1,t+1,p=-)$

13. $\Box(t,t,compressed) \wedge \Diamond(t,t,\neg v=+) \supset \Box(t+1,t+1,compressed)$

No change in velocity unless the acceleration is non-zero:

14. $\Box(t,t,v=0) \wedge \Diamond(t,t,\neg a=-) \wedge \Diamond(t,t,\neg a=+) \supset \Box(t+1,t+1,v=0)$

15. $\Box(t,t,v=-) \wedge \Diamond(t,t,\neg a=-) \wedge \Diamond(t,t,\neg a=+) \supset \Box(t+1,t+1,v=-)$

16. $\Box(t,t,v=+) \wedge \Diamond(t,t,\neg a=-) \wedge \Diamond(t,t,\neg a=+) \supset \Box(t+1,t+1,v=+)$

No change in acceleration unless the force is non-zero:

17. $\Box(t,t,a=0) \wedge \Diamond(t,t,\neg f=-) \wedge \Diamond(t,t,\neg f=+) \supset \Box(t+1,t+1,a=0)$

18. $\Box(t,t,a=-) \wedge \Diamond(t,t,\neg f=-) \wedge \Diamond(t,t,\neg f=+) \supset \Box(t+1,t+1,a=-)$

19. $\Box(t,t,a=+) \wedge \Diamond(t,t,\neg f=-) \wedge \Diamond(t,t,\neg f=+) \supset \Box(t+1,t+1,a=+)$

Causal rules

When released, the block will have a negative force:

20. $\Box(t,t,f=0) \wedge \Box(t,t,released) \wedge \Diamond(t,t,\neg springbreak) \supset \Box(t+1,t+1,f=-)$

If the spring breaks, its force will be zero

21. $\Box(t,t,\text{springbreak}) \supset \Box(t+1,t+1,f=0)$

Once the spring breaks, it remains broken unless fixed:

22. $\Box(t,t,\text{springbreak}) \wedge \Diamond(t,t,\neg\text{fixed}) \supset \Box(t+1,t+1,\text{springbreak})$

Hooke's law: $F = kx$

23. $\Box(t,t,\text{stretched}) \wedge \Diamond(t,t,\neg\text{held}) \wedge \Diamond(t,t,\neg\text{springbreak}) \supset \Box(t+1,t+1,f=-)$

24. $\Box(t,t,p=+) \wedge \Diamond(t,t,\neg\text{held}) \wedge \Diamond(t,t,\neg\text{springbreak}) \supset \Box(t+1,t+1,f=-)$

25. $\Box(t,t,p=0) \wedge \Diamond(t,t,\neg\text{held}) \wedge \Diamond(t,t,\neg\text{springbreak}) \supset \Box(t+1,t+1,f=0)$

26. $\Box(t,t,p=-) \wedge \Diamond(t,t,\neg\text{held}) \wedge \Diamond(t,t,\neg\text{springbreak}) \supset \Box(t+1,t+1,f=+)$

27. $\Box(t,t,\text{compressed}) \wedge \Diamond(t,t,\neg\text{held}) \wedge \Diamond(t,t,\neg\text{springbreak}) \supset \Box(t+1,t+1,f=+)$

Newton's law: $F = ma$

28. $\Box(t,t,f=+) \supset \Box(t+1,t+1,a=+)$

29. $\Box(t,t,f=0) \supset \Box(t+1,t+1,a=0)$

30. $\Box(t,t,f=-) \supset \Box(t+1,t+1,a=-)$

Transposition: rules for moving along the positions stretched, $+, 0, -$, compressed.

31. $\Box(t,t,\text{stretched}) \wedge \Box(t,t,v=-) \wedge \supset \Box(t+1,t+1,p=+)$

32. $\Box(t,t,p=+) \wedge \Box(t,t,v=-) \wedge \supset \Box(t+1,t+1,p=0)$

33. $\Box(t,t,p=+) \wedge \Box(t,t,v=+) \wedge \supset \Box(t+1,t+1,\text{stretched})$

34. $\Box(t,t,p=0) \wedge \Box(t,t,v=-) \wedge \supset \Box(t+1,t+1,p=-)$

35. $\Box(t,t,p=0) \wedge \Box(t,t,v=+) \wedge \supset \Box(t+1,t+1,p=+)$

36. $\Box(t,t,p=-) \wedge \Box(t,t,v=-) \wedge \supset \Box(t+1,t+1,\text{compressed})$

37. $\Box(t,t,p=-) \wedge \Box(t,t,v=+) \wedge \supset \Box(t+1,t+1,p=0)$

38. $\Box(t,t,\text{compressed}) \wedge \Box(t,t,v=+) \wedge \supset \Box(t+1,t+1,p=-)$

From the definition of velocity in terms of acceleration:

39. $\Box(t,t,v=-) \wedge \Box(t,t,a=-) \supset \Box(t+1,t+1,v=-)$

40. $\Box(t,t,v=-) \wedge \Box(t,t,a=+) \supset \Box(t+1,t+1,v=0)$

41. $\Box(t,t,v=0) \wedge \Box(t,t,a=-) \supset \Box(t+1,t+1,v=-)$

42. $\Box(t,t,v=0) \wedge \Box(t,t,a=+) \supset \Box(t+1,t+1,v=+)$

43. $\Box(t,t,v=+) \wedge \Box(t,t,a=-) \supset \Box(t+1,t+1,v=0)$

44. $\Box(t,t,v=+) \wedge \Box(t,t,a=+) \supset \Box(t+1,t+1,v=+)$

Result The cmi model is computed for the cases where

1. the spring never breaks
2. the spring breaks at time-point 5 with the axiom $\Box(5,5,\text{springbreak})$

The results are listed in Table 3.1. The table shows the position, velocity, acceleration, and force of the block at each time-point from 1 to 18.

The overall behaviour of the block in each case is:

Case1: The block oscillates between *stretched* and *compressed*

Case2: The block moves towards the wall and stops at the position *compressed*

Table 3.1: The result of computation for the simple oscillator problem.

time	Case1				Case2			
	position	v	a	f	position	v	a	f
1	stretched	0	0	0	stretched	0	0	0
2	stretched	0	0	0	stretched	0	0	0
3	stretched	0	0	0	stretched	0	0	0
	released				released			
4	stretched	0	0	−	stretched	0	0	−
5	stretched	0	−	−	stretched	0	−	−
					springbreak			
6	stretched	−	−	−	stretched	−	−	0
7	+	−	−	−	+	−	0	0
8	0	−	−	−	0	−	0	0
9	−	−	−	0	−	−	0	0
10	compressed	−	0	−	compressed	−	0	0
11	compressed	−	+	−	compressed	−	0	0
12	compressed	−	+	+	compressed	−	0	0
13	compressed	+	+	+	compressed	−	0	0
14	−	+	+	+	compressed	−	0	0
15	0	+	+	+	compressed	−	0	0
16	+	+	+	0	compressed	−	0	0
17	stretched	+	0	−	compressed	−	0	0
18	stretched	+	−	−	compressed	−	0	0

In Case2, the block begins to move towards the wall when the spring breaks; no more force being applied from the spring, the block remains still at the *compressed* position. Since there is no knowledge in the theory about the position closer to the wall other than *compressed*, the block might keep on moving towards the wall (with negative velocity) until it reaches there.

This result illustrates that as far as the overall movement of the block is concerned, the formulation using CI appears to produce a simulated behaviour. However, upon closer examination, it is revealed that it is not a physical model of the oscillator. For instance, from Newton's Law ($F = ma$), f and a should always have the same sign (since it is always the case that $m > 0$). However, in Table 3.1, there are states in which the signs of f and a do not match. What the result is actually modelling is not the physics of the oscillator, but the flow of information where a change in a would cause a change in f; in physics, this does not involve temporal delay. Such simultaneous changes are not handled properly in CI. This is discussed further in Section 3.8.1.

3.7.2 RC current divider

The *envisioning* of the behaviours of simple electrical circuits is another popular problem in qualitative reasoning. Reasoning about the directions of the electrical currents, voltages, whether the light bulbs are on/off can be done to a considerable extent solely by examining their causal relations and without precise numerical calculations. The following problem is introduced by Williams as an example of qualitative reasoning [Williams 84]. The scenario follows the way the human expert thinks about a RC current divider circuit illustrated in Figure 3.3.

Scenario A typical explanation of the RC current divider circuit is:

> *When the input current becomes positive, the capacitor initially acts like an incremental short and all the current goes into the capacitor. As the capacitor charges, this produces a positive voltage across the resistor, causing* $I(R1)$ *to be positive.*

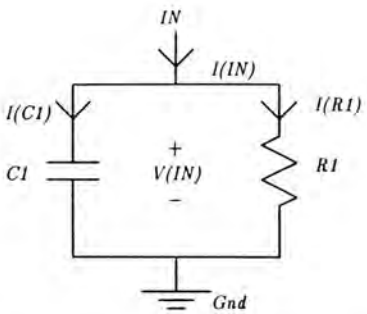


Figure 3.3: The RC current divider circuit.

To make the problem more interesting, the input current is assumed to be increased at time points 3 and 20, and potentially stay positive for a time interval of 10 after each increase.

Formulation An first-order inertial theory looks as follows:

Boundary conditions

The input current is increased at time points 3 and 20

1. □(3,3,increase(current,in))
2. □(20,20,increase(current,in))

The input current is potentially 0 and the capacitor is potentially discharged

3. POTEN(1,∞,p-current(in,0))
4. POTEN(1,∞,p-discharged(c))

Projection

5. $\forall C$ PROJECT(t_1,p-charged(C),t,t,charged(C))

6. $\forall C$ PROJECT(t_1,p-discharged(C),t,t,discharged(C))

7. $\forall C$ PROJECT(t_1,p-charging(C),t,t,charging(C))

8. $\forall D \forall S$ PROJECT(t_1,p-current(D,S),t,t,current(D,S))

9. $\forall D \forall S$ PROJECT(t_1,p-voltage(D,S),t,t,voltage(D,S))

Causal rules

- Input current

Each time the input current is increased, it will potentially stay positive for time 10, and is no longer 0

10. $\Box(t,t,\text{increase(current,in)}) \wedge \Diamond(t,t,\neg\text{current(in,+)})$
$\supset$ POTEN(t+1,t+10,p-current(in,+))

11. $\Box(t,t,\text{increase(current,in)}) \wedge \Box(t,t,\text{current(in,0)}) \supset \Box(t+1,t+1,\neg\text{current(in,0)})$.

- Capacitor

When current flows in, the capacitor will begin to be charged; when it begins to charge, the state is charging and the state discharged terminates

12. $\Box(t,t,\text{current}(c,+)) \wedge \Diamond(t,t,\neg\text{charging}(c)) \wedge \Diamond(t,t,\neg\text{charged}(c))$
$\wedge \Diamond(t,t,\neg\text{begin-charge}(c)) \supset \Box(t+1,t+1,\text{begin-charge}(c))$

13. $\Box(t,t,\text{begin-charge}(c)) \supset \Box(t+1,t+1,\text{charging}(c))$

14. $\Box(t,t,\text{begin-charge}(c)) \wedge \Box(t,t,\text{discharged}(c)) \supset \Box(t+1,t+1,\neg\text{discharged}(c))$

If charging at time t, then it will be potentially charged from t+1 to infinity

15. $\forall C\ \Box(t,t,\text{charging}(C)) \supset$ POTEN(t+1,∞,p-charged(C))

When capacitor is not charged, it will act as an incremental short so that the input current will flow into it

16. $\Box(t,t,\text{current}(in,+)) \wedge \Diamond(t,t,\neg\text{charged}(c)) \supset \Box(t+1,t+1,\text{current}(c,+))$

17. $\Box(t,t,\text{current}(in,+)) \supset \Box(t+1,t+1,\neg\text{current}(c,0))$

When the capacitor is charged, the voltage will be positive and is no longer 0

18. $\Box(t,t,\text{charged}(c)) \wedge \Box(t,t,\text{voltage}(v,0)) \supset \text{POTEN}(t+1,\infty,\text{p-voltage}(v,+))$

19. $\Box(t,t,\text{charged}(c)) \supset \Box(t+1,t+1,\neg\text{voltage}(v,0))$

When the capacitor is charged and the input current stops, the capacitor will start to discharge and is no longer charged

20. $\forall C\ \Box(t,t,\text{charged}(C)) \wedge \Diamond(t,t,\neg\text{current}(in,+)) \supset \Box(t+1,t+1,\text{discharge}(C))$

21. $\forall C\ \Box(t,t,\text{charged}(C)) \wedge \Diamond(t,t,\neg\text{current}(in,+)) \supset \Box(t+1,t+1,\neg\text{charged}(C))$

If a capacitor discharges, it will potentially be discharged

22. $\forall C\ \Box(t,t,\text{discharge}(C)) \wedge \Diamond(t,t,\neg\text{discharged}(C)) \supset \text{POTEN}(t+1,\infty,\text{p-discharged}(C))$

23. $\forall C\ \Box(t,t,\text{discharge}(C)) \wedge \Diamond(t,t,\neg\text{discharged}(C)) \supset \Box(t+1,t+1,\neg\text{charged}(C))$

When discharged, the voltage will be 0 and remains the same

24. $\forall C\ \Box(t,t,\text{discharged}(C)) \wedge \Diamond(t,t,\neg\text{voltage}(v,0)) \wedge \Diamond(t,t,\neg\text{current}(in,+)) \supset \text{POTEN}(t+1,\infty,\text{p-voltage}(v,0))$

25. $\Box(t,t,\text{discharged}(c)) \wedge \Diamond(t,t,\neg\text{current}(in,+)) \supset \Box(t+1,t+1,\neg\text{voltage}(v,+))$

- Resistor

Positive voltage will cause the flow of the current through the resistor

26. $\Box$(t,t,voltage(v,+)) $\wedge$ $\Box$(t,t,current(r,0)) $\supset$ POTEN(t+1,∞,p-current(r,+))

27. $\Box$(t,t,voltage(v,+)) $\supset$ $\Box$(t+1,t+1,$\neg$current(r,0))

When the voltage is 0 the current at the resistor will potentially be 0

28. $\Box$(t,t,voltage(v,0)) $\wedge$ $\Diamond$(t,t,$\neg$current(r,0)) $\supset$ POTEN(t+1,∞,p-current(r,0))

29. $\Box$(t,t,voltage(v,0)) $\wedge$ $\Box$(t,t,current(r,+)) $\supset$ $\Box$(t+1,t+1,$\neg$current(r,+))

When the voltage becomes positive/negative then the current at the resistor will be positive/negative

30. $\forall$ S $\Box$(t,t,voltage(v,S)) $\wedge$ $\Box$(t,t,current(r,0)) $\supset$ POTEN(t+1,∞,p-current(r,S))

Result The result for the computation for the cmi model of the theory described above is summarised in Table 3.2. The duration of the inference is from $t = 1$ to $t = 30$.

The input current remains positive for time-interval of 10 from time 3 which charges the capacitor; when the input current is shut off, the capacitor begins to discharge until the current increases again at time 20. Undoubtedly this is the intended behaviour of the circuit for the set-up given.

3.8 Evaluation

The results for the sample problems above support the view that computed "unique cmi models" are indeed the intended models for the theories. Moreover, the predicted behaviour of a theory changes as intended in accordance with the different set-ups.

In the course of formulating and solving the sample problems above, however, two notable issues arose: one is the interpretation of the time lag between the events which, intuitively, should happen simultaneously; the other is the treatment of the semantic information of predicates that are independent of time. The difficulties created by

Table 3.2: The result of computation for the RC circuit divider problem.

time	I(IN)	I(C)	Capacitor	V(R)	I(R)
1	0	—	discharged	—	—
2	0	—	discharged	0	—
3	0	—	discharged	0	0
	increase				
4	+	—	discharged	0	0
5	+	+	discharged	0	0
6	+	+	discharged	0	0
			begin-charge		
7	+	+	charging	0	0
8	+	+	charged	0	0
9	+	—	charged	+	0
10	+	—	charged	+	+
11	+	—	charged	+	+
12	+	—	charged	+	+
13	+	—	charged	+	+
	The input current is shut off				
14	—	—	charged	+	+
15	—	—	discharge	+	+
16	—	—	discharged	+	+
17	—	—	discharged	0	+
18	—	—	discharged	0	0
19	—	—	discharged	0	0
20	increase	—	discharged	0	0
21	+	—	discharged	0	0
22	+	+	discharged	0	0
23	+	+	discharged	0	0
			begin-charge		
⋮	The same sequence as 7 to 12				
30	+	—	charged	+	+

these issues are avoidable without radical change of the logic CI, as has been done in the formulation above.

The significance of the extension to the first order case is that it reduces the number of axioms in a theory to a considerable extent. This is a general advantage of the first order logic to the propositional which enhances the expressiveness of the language.

3.8.1 Mythical time

The logic CI does not allow the cause and the effect to occur at the same time point, i.e., practically, the *ltp* of the antecedent of an axiom must be smaller than that of the consequent. In terms of causality, this is a plausible restriction since the effect should follow the cause. There are, however, cases when one event, intuitively, simultaneously "follows" another.

Such is the interpretation of physical equations. For example, in the "simple oscillator problem" above, there are physical equations such as Hooke's law and Newton's law. Of these, Newton's law

$$F = ma$$

where F is the force, m the mass of the object, and a its acceleration, describes the *covariance* between the parameters, rather than their causal relationship. The acceleration a is expected to change simultaneously with the change in the force. Since there is no way but to express this relation in terms of causal relationships, it is formulated as axioms 28 to 30 (p. 52).

These axioms are the interpretation of a physical equation as a causality. In this interpretation, the cause and the effect depend on the context or the scenario of the reasoning. It is analogous to the transformation of an equation in terms of a parameter, which needs to be calculated with other variables of known value.[11] In other words, the equation by itself may not specify the causal relationships between the parameters, but

[11] To illustrate this situation, consider a problem in geometry about a circle: given the formula $S = \pi r^2$, where S is the area of a circle and r its radius. If we were to calculate the value of r given the area of a circle, the equation is transformed to $r = \sqrt{\frac{S}{\pi}}$. In this case, the area can be considered to be the *cause* for the *effect* of obtaining the value of its radius.

can be interpreted as causal relations. In the case of the "simple oscillator problem", Newton's law is interpreted as the force being the cause and the acceleration as the effect. This idea of identifying the cause and effect among physical equations depending on the means and the ends on obtaining a desired flow of parameter assignment can be seen in the work on causal ordering by Iwasaki and Simon[86].

This explanation sounds reasonable. But should there nevertheless be a time point difference between the *cause* and the *effect*? Should it not happen simultaneously? We can think of the time point difference as an infinitesimal time interval, which is *almost* 0. The *actual* length of time is irrelevant to the time-points. That is, although there is a partial ordering between the time points, the events at these time-points happens, effectively, at the same time. I call this *mythical time (interval)*.

What happens when an event happens during a *mythical time*? In the sample problem "simple oscillator", if the event "springbreak" occurs at time point 1, the computed cmi model will be such that the block remains at the initial position although it was released in the previous time-point. The spring broke while the value was being propagated from the force to the velocity via acceleration, and the velocity never changed from 0. It might seem unreasonable at first, but if we take the *actual* length of time during a couple of time-points as 0, this result turns out to be plausible. After all, it is the human who interprets the time-points and their intervals.

3.9 Summary

In summary, this chapter introduced the idea about chronological minimisation as advocated by Shoham, and extended the logic CI to the first order case to enhance its expressiveness. The major points are:

- Shoham's logic CI provides sound and computational framework for reasoning about change based on the preference criterion that the deflection happens as late as possible.

- The extension of logic CI to the first-order case enhanced its expressiveness at the cost of computational complexity, but in practice it can be implemented relatively efficiently.
- Unique cmi model theorem was applied to some qualitative reasoning problems and exhibited that it indeed provided a most likely solution to the problems.

In the next chapter, we extend the use of logic CI further to investigate its applicability for reasoning about the past through abduction.

Chapter 4

Reasoning about past events

Reasoning about change has been an issue which has drawn the attention of many AI researchers. In the mainstream of this field are prediction tasks, which reason about future events given a set of events in the present. As we have seen so far, prediction tasks and reasoning about change involve problems such as the ramification problem, and various formalisms were devised to cater for these problems.

While causal theories are capable of providing plausible predictions, less emphasis has been placed on reasoning about the past. Questions such as 'what sequence of events led up to these events?' have not been thoroughly explored. Also the effort involved in creating such causal theories requires a careful construction and assessment, and it is virtually impossible to write down every single causal rule in the domain; there is always room for unexpected omissions. This is less of a problem when dealing with a prediction task, since the best we can do is to anticipate possible outcomes by the knowledge provided. However, when we attempt to make a query about the past, the lack of causal knowledge can easily result in 'unknown' answers. Looking into the future without adequate causal knowledge merely concludes that nothing happens in the future; the same case into the past cannot explain why the present events occurred. A better approach is to explore the possible causal connection between events to assume the cause(s) for a given event, and verify it in some acceptable fashion.

Chapter 3 discussed the prediction task: in other words, reasoning about events in the future. Shoham's logic CI [Shoham 88] was advocated for this task: it identified the

most plausible prediction based on the notion of persistence and minimal deflection over time. This chapter investigates methods to reason about the past. This task, which I shall term *reconstruction*[1], is to give a plausible sequence of events which led up to the events in the present.

Essentially, we attempt to deal with this problem in the following manner. For a causal theory Ψ as defined in Chapter 3, we define a set (which is called a bci set), that expresses what would be true if the notion of backward persistence is used, together with some heuristics where there are choices.[2] Given the bci set, and the cmi model, a non-deterministic procedure abduces some extra statements which we add to Ψ to get Ψ'; the Ψ' obtained is a causal theory for which the bci set and the cmi model coincide.

The aim of dealing with the reconstruction task in this context, as is discussed later in Chapter 7, is to investigate its possible application to the synthesis of causal rules. While prediction provides the most likely outcome, or simulation, of the device, reconstruction would provide suggestions concerning 'how to achieve the necessary outcome', i.e., the emphasis is placed on not only 'what' is achieved, but 'how' it is achieved. In the field of design, Iwasaki and Chandrasekaran[92] emphasises the significance of specifying 'how to' achieve a function of a device in the design verification tasks.

4.1 Reconstruction task

The nature of the problem we deal with here is *reconstruction*, i.e., given a present state/condition, find out a plausible history in the past. We may or may not have all the information about the present state, or the conditions in the past. The knowledge we would have is that used in causal reasoning for prediction, such as causal rules, potential histories and boundary conditions. Reconstruction is a task which is as commonly performed as prediction in our everyday life. For example, if there is a book lying open on your desk and you had not left it yourself, you would think that there was somebody sitting there before you arrived. At the same time, if you found your

[1]This term was suggested by Alan Smaill.

[2]This set is not in general a model of Ψ.

favourite pen missing from your pen stand, you would think that the same person has taken it away.

As in a prediction task, a reconstruction task involves facts and assumptions about the current state, the knowledge about the domain, and causal theories. Formally, a reconstruction task can be defined as follows. Let C be a set of events known to be true between a range of time points. If t_p is the *earliest time point* (etp) among C, a *reconstruction task* is to find out a plausible occurrence of events at t such that $t < t_e$, given an appropriate knowledge about the course of nature.

Definition 12 (Earliest time points) *The* earliest time point (etp) *of a base formula is the earliest time point mentioned in it, formally defined as follows. Let* $eval(trm)$ *stand for the evaluation of* trm*, which is an integer time point.*

1. *The* etp *of* $\text{TRUE}(trm_a, trm_b, r(trm_1, ..., trm_n))$ *(where* trm_a *and* trm_b *are temporal terms,* $trm_1, ..., trm_n$ *are non-temporal terms, and* r *is a relation symbol) is* $min\{eval(trm_a), eval(trm_b)\}$.
2. *The* etp *of* $\varphi_1 \wedge \varphi_2$ *is the earliest (w.r.t. the standard interpretation of time) between the* etp *of* φ_1 *and the* etp *of* φ_2.
3. *The* etp *of* $\neg\varphi$ *is the* etp *of* φ.

The etp of a causal theory Ψ *is the earliest among all the base formulae appearing in it.*

Given a causal theory, this task is achieved initially by performing abduction over causal rules. This section includes the discussion concerning the importance of persistence backwards in time and how it should be handled.

4.1.1 Past events

There are a number of benchmark problems for temporal/causal reasoning languages, one of which is the Yale Shooting Problem (YSP). The following is an example of

the problem, with some technical modifications from the original [Kautz 86], which involves reasoning about past events from the present observations.

Example 2 (Stolen car problem (SCP)) *The initial conditions ($t = 1$) are 1) leaving a car in the parking lot, and 2) the car is initially not stolen. The evidence is that you find that the car is stolen at $t = 10$. The task is to infer the time the theft took place.*

In the same paper Kautz advocates the logic of persistence, the framework to treat forward persistence of events in a formal manner. SCP was introduced in the final part of the paper to address the limitation of his approach when applied to reasoning about past events. While addressing this problem in the context of a forward minimisation approach, Kautz states:

> Using the techniques [of forward persistence] just described, I can infer that the car was in the parking lot up to the shortest possible time before I knew it was gone. This is clearly an unreasonable inference. Someone could have stolen it five minutes after I left it there; I have no reason to prefer an explanation in which it vanished five seconds before I glanced out my office window.

As far as this problem is concerned, there seems to be no preferred explanation. All we can say about when the car was stolen is that it happened 'sometime' between $t = 1$ and $t = 10$. In this case the preference for persistence means very little. This problem is, however, a very difficult one to deal with in the first place; there is too little information. When we are confronted with such a problem in real life, we can always speculate what happened. And as long as the speculation lies within some reasonable range, there is no way we can prove or deny it with such limited information. As we accumulate more information we can find out more about, or narrow down, the time range. For instance, if it was raining, the surface of the parking space where the car was parked would be drier if it was stolen just before we found it gone. There might be some witnesses who can provide more information. Or it might just have been towed

away from the "no parking" area; if so, it is likely that it has been some time after you left the car since it would take time for the traffic warden to call the towing car.[3]

One obvious way to tackle the reconstruction problem is to use a causal theory which is designed to make causal statement about the past from the present. A notable attempt to represent causal knowledge in a cause-oriented fashion is that of MYCIN [Shortliffe 76] where the rules are used to infer the cause from effect. This inverted implication in causal rules of MYCIN suits its specific purpose of identification of the diseases from symptoms. However, as Shanahan points out in the context of the explanation task [Shanahan 89], the formulation of inverted implications from causal laws is counter-intuitive and restricted to the task in which it is possible or appropriate. In other words, this approach is not appropriate for most tasks dealing with causation where it is more intuitive to assume forward formulation of causation. Also MYCIN rules do not carry the notion of sequence of events, which is a vital aspect in reasoning about temporal relations.

Another aspect of this problem which makes it difficult to deal with the similar approach of forward persistence is that the chronological sequence is 'closed' at both ends. For a prediction task, the problem is to find out the events in the time points which are later than any time point in the causal theory. In other words, it is *unbounded* in the future and any event can happen as long as it follows the causal theory. In the case of SCP, the problem is not only reconstruction, but *interpolation*. It is *bounded* at some points in the past, i.e., there is a fixed boundary condition in the past—in this case that the car was parked. Without this boundary condition, the reconstruction problem may be able to be handled as it was for the prediction, that is to assert 'the car had never been there.' This assertion, however, is in most cases not plausible. From a retrospective point of view, where we view from the effect to the cause, it is reasonable to expect that an event should exist which resulted in the present state of the world.[4] On the other hand, our speculation about the future seems to be more tolerant in the

[3] Another way to look at this problem is that there is no point in reasoning about the absolute time from the given information; the time points merely denotes the temporal order of events and not necessarily the duration of time. In other words, time lapsed between time points $t = 1$ and $t = 7$ could be 'shorter' than that between $t = 8$ and $t = 9$.

[4] The ultimate task of this belief is the quest for the origin of the universe.

sense that we allow certain events to last for ever. For instance, an old Scottish stone house may stand for centuries to come, and we are not always sure that it will ever be torn down.

In looking for a plausible reconstruction, an important idea is the persistence of states in both directions, forward and backward. From this view point, when there is an isolated event which is known to persist over time, it is plausible to infer from this observation that the event has been, and will continue to be, taking place until and from that point. This can be illustrated analogously to a stone thrown into the water creating a ripple propagating in every direction. The stone here is the event observed; the propagation of the wave is the expected persistence of the event. If there is something which blocks the propagation of the ripple, it affects the undisturbed extension of the wave, just as an event that clips the persistence changes the direction of the event trajectory. Furthermore, the wave eventually reaches the edge of the pool, which can be seen as the boundary conditions.

In the later section, a system based on CI is utilised to reason about events backwards in time. Intuitively, if there is knowledge that an event P potentially persists from time t_1 to t_2, we prefer abductions in which this persistence not only applies forward in time, but backward as well. First a brief description of interpolation problems is given.

4.2 Interpolation problems

When there are sets of events known to occur, or have occurred, at different time points, and if these sets of events are expected to have causal relations, one may wonder what happened in between to link these events. While a reconstruction problem deals with the events which precede *all* the known events, it does not care what happened in between such events. These events are those dealt with by interpolation. Figure 4.1 illustrates the relationship between prediction, reconstruction and interpolation.

Formally, given events $\phi(t_1)$ and $\phi(t_2)$, where $t_1 < t_2$, where there are no obvious causal links between $\phi(t_1)$ and $\phi(t_2)$, an interpolation problem is to find a set of events

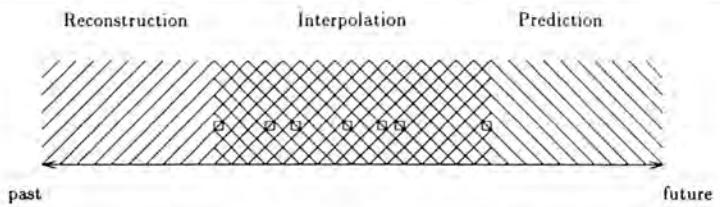


Figure 4.1: Prediction, reconstruction and interpolation. Boxes indicate the known events. Notice that the region concerned in prediction tasks is that after the latest event known. Similarly, reconstruction tasks deals with the region before the earliest known event. Interpolation tasks are performed over the region between the earliest and the latest known events.

between these time points, $\phi(t_i)$ such that $t_1 < t_i < t_2$, which together with existing causal theory Ψ constructs a model M which is consistent with $\phi(t_1)$ and $\phi(t_2)$. That is,

$$\Psi \cup \{\phi(t_i) : t_1 < t_i < t_2\} \models M \tag{4.1}$$

for all t_i which are $t_1 < t_i < t_2$. Hereafter, the span of time points from t_1 to t_2 is referred to as the *range* of an interpolation problem, and $\phi(t_1)$ and $\phi(t_2)$ its *initial* and *terminal* set, respectively.

4.3 Abduction over causal rules

Given a sound and consistent causal theory,[5] there is a unique model which is most preferred in terms of chronological ignorance. That is, it prefers a model in which possible divergence from the potential history is delayed as long as possible. This clearly identifies the most plausible outcome in the future, i.e., emi model forward in time.

Having focused on the identification of the most plausible model forward in time, we

[5] A *sound* causal theory does not contain contradictory defaults ($\diamond$-conditions); a *consistent* theory does not contain two causal rules which have the same antecedent but contradictory consequents. (See Section 3.4

are now interested in using the causal theory in order to reason backward in time, i.e., what can we conclude about the past given the causal knowledge. This process is considered to be *abduction* over causal rules. The term *abduction* was first used by C. W. Peirce to refer to the method of reasoning which draws conclusions about the cause for an event [Peirce 23]. The example below illustrates how abduction may be performed in the syntax of causal theory.

Example 3 (Glass window problem) *Assume a causal theory which comprises the following causal rules.*[6]

1. $\Box(t,t,\textit{glass-window}) \wedge \Box(t,t,\textit{vandalism}) \supset \Box(t+1,t+1,\textit{break-glass})$.
2. $\Box(t,t,\textit{glass-window}) \wedge \Box(t,t,\textit{accident}) \supset \Box(t+1,t+1,\textit{break-glass})$.
3. $\Box(t,t,\textit{glass-window}) \wedge \Box(t,t,\textit{gale-force-wind}) \supset \Box(t+1,t+1,\textit{break-glass})$.
4. $\Box(t,t,\textit{break-glass}) \supset \text{POTEN}(t+1,\infty,\text{p-}\textit{broken-glass})$.
5. $\Box(t,t,\textit{broken-glass}) \wedge \Box(t,t,\textit{replace-glass}) \supset \text{POTEN}(t+1,\infty,\text{p-}\textit{glass-window})$.
6. $\text{POTEN}(1,\infty,\text{p-}\textit{glass-window})$.

When the event *break-glass* is observed at time t_1, given the causal theory above, we can assume three possible sets of events which might have happened at time $t_1 - 1$, *viz.*, {*glass-window, vandalism*}, {*glass-window, accident*}, or {*glass-window, gale-force-wind*}. There is no preferred event among these causes and they are, logically speaking, equally possible. Moreover, any union of these sets is also possible.[7]

If the event *broken-glass* is observed at time t_2, we can apply the inertial rule 4 backwards to consider that the event *break-glass* should have occurred sometime earlier, say at $t_3 (t_3 < t_2)$, and the events *broken-glass* had persisted between t_3 and t_2. So far so good.

---

[6] Technically, we also require rules that clip persistence, e.g., $\Box(t,t,\textit{glass-window}) \wedge \Box(t,t,\textit{vandalism}) \supset \Box(t+1,t+1,\neg\textit{glass-window})$, for rules 1,2,3 and 5. However, for simplicity, we assume *break-glass* to be contradictory to *glass-window*.

[7] There can be more than one cause for an effect. For instance, both *gale-force-wind* and *accident* might have happened at time $t_1 - 1$ which resulted in causing *break-glass* at the same time.

What if we observe the event *glass-window* at time t_4? There are two cases. The first case is that *glass-window* has been true since the initial time point 1 (rule *6*). The second case is that the events *break-glass* and *replace-glass* occurred in between (See Figure 4.2).

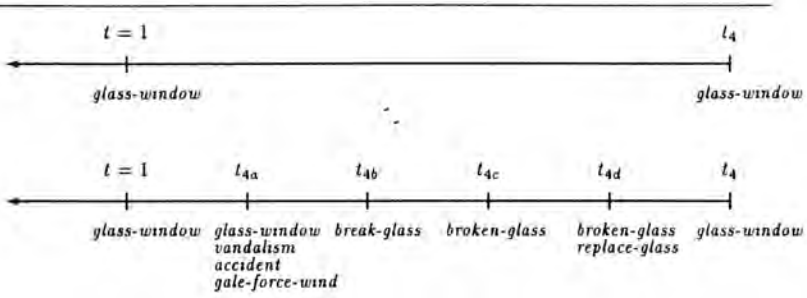


Figure 4.2: Was the glass broken?—The direction of the arrow, from the present (t_4) to the past, indicates the direction of reasoning.

In cases such as the one just examined, when there is a knowledge that the events observed persist for a period of time, it is plausible to prefer the case where there is a smaller number of events. In this case, the first case is preferred to be the likely sequence of past events. If something were to be there, it is more parsimonious to think that it was there ever since it was placed. This is analogous to applying the chronological minimisation backwards in time; nothing happened unless we know that it happened. Naturally, if we acquire the knowledge that there was a point in the past in which the vandalism towards the glass window took place, the history in which nothing happened no longer holds.

The default conditions can be dealt with in the following way. Suppose rule *1* had the default condition that the glass window was not a toughened glass which doesn't break by vandalism. The rule then looks like

1a. $\Box(t,t,glass\text{-}window) \wedge \Box(t,t,vandalism) \wedge \Diamond(t,t,\neg toughened\text{-}glass)$
$\supset \Box(t+1,t+1,break\text{-}glass).$

Given *break-glass* at time t_5, we abduce {*glass-window, vandalism*} at time $t_5 - 1$ provided that we do not know, or prove, *toughened-glass* at time $t_5 - 1$. This is different from the case where the rule is

1b. $\Box(t,t,glass\text{-}window) \wedge \Box(t,t,vandalism) \wedge \Box(t,t,\neg toughened\text{-}glass) \supset \Box(t+1,t+1,break\text{-}glass)$.

when we abduce {*glass-window, vandalism,* $\neg$*toughened-glass*}, provided that none of the three events contradicts with the known events.

4.4 Abduction with backward persistence

When there is a state which is likely to persist, the preference here is given to those sets with the least number of changes, chronologically backwards. That is, given an event p, which constitutes a potential history, at time t, we assume, unless explicitly told otherwise, that p was there at time $t - 1$.

This section attempts to construct a framework for reasoning abductively backwards in time, using backwards chronologically ignorant sets (*bci* sets). First, the preference for backward persistence is described. The focus is whether it is possible to use the existing framework to represent inference into the past. Second, I attempt to formulate an algorithm to compute the bci set based on the language CI. The bci set is supposed to describe the sequence of events that led to the given event(s) with the smallest number of deflections as possible.

This is an attempt to chronologically minimise the number of events (or deflections) occurring in the set. As we see later, chronological minimisation backwards in time in fact supports the *minimality of causes* as defined by Cox and Pietrzykowski[86]. Preference of persistence guarantees the minimality of causes, since anything more than what has occurred will at least increase the number of causes by one.

4.4.1 Backward projection

The syntax for causal rules and causal theory remains unaltered. The definitions of *ltp* (latest time point) and consistency, soundness conditions also still hold. The only difference is what is represented by the persistence condition. We can interpret persistence conditions as the general properties of persistence about events, which implies that the persistence not only holds forward but also backward. For instance, the inertial rule POTEN$(t_1, t_2, \mathrm{p}{-}P)$, would literally mean that the event P persists in *both* directions (forward and backward) between t_1 and t_2 regardless of which time point to start from.

In order to make this backward persistence work, we need a way to interpret inertial rules which project the persistence of events backward. In addition to the forward projection

$$\text{PROJECT}(t_1, \mathrm{p}{-}P, t_2, t_3, [\neg]P) =_{def} (\exists v(t_3 \leq v \wedge \Box(t_1, v, \mathrm{p}{-}P))) \supset \Box(t_2, t_3, [\neg]P) \tag{4.2}$$

such that $t_1 \leq t_2 \leq t_3$ and $\mathrm{p}{-}P$ denoting a proposition P for a potential history, along with soundness and consistency conditions, we now need the backward projection of the form

$$\text{B-PROJECT}(t_1, \mathrm{p}{-}P, t_2, t_3, [\neg]P) =_{def} (\exists v(v \leq t_1 \wedge \Box(v, t_3, \mathrm{p}{-}P))) \supset \Box(t_1, t_2, [\neg]P). \tag{4.3}$$

B PROJECT can be used to project events defined as a potential history backward in time. It is important to note that we are not creating a causal rule with inverted implication. This backward projection applies to the inertial rules in causal theories, but states that the inertia can be assumed backwards in time. To make the distinction clear, we will identify forward and backward projection as F PROJECT and B-PROJECT respectively.[8] By convention, hereafter we write PROJECT to mean that it is both

[8] This is to cater for the case where some states only persist in one direction.

F-PROJECT and B-PROJECT.

4.4.2 Preference of persisting states and minimal causes

As stated earlier, if there is an event which potentially persists backwards, the preference is given to the persistence of such an event unless some conditions prevent it. However, if there is an event which is known to have happened, and has possibly given rise to the potential history, the persistence could be truncated and conclude that the potential history has initiated at that point. For instance, in the Glass Window Problem, if it was known that the event *replace-glass* took place at t_2, between $t = 1$ and t_4, then it is more likely that the glass was broken prior to t_2 and the potential history of *glass-window* was initiated by the *replace-glass* event, rather than to pursue the backward persistence of *glass-window*.

To summarise, the candidates of abduction are obtained by the following procedures:

1. If there is a boundary condition at time t which is (a part) of the necessary conditions for a causal rule which has as its consequent the current event, abduce the events in the necessary conditions provided that no such event gives rise to contradictory assertions (i.e., contradiction with other boundary conditions).

2. If the current event p is a part of a potential history which may persist backwards, and if the persistence of the event does not bring about contradictory assertions, let p persist backwards.

Among these candidates, the preference is given to those which abduce the least number of necessary conditions.

It is important to note that when backward persistence is not chosen, the abduction would prefer as small number of events to have occurred as possible. For instance, given an event p at time $t_a + 1$, and two causal rules

$A.\ \Box(t,t,a) \wedge \Box(t,t,b) \wedge \Box(t,t,c) \supset \Box(t+1,t+1,p)$

$B.\ \Box(t,t,d) \wedge \Box(t,t,f) \supset \Box(t+1,t+1,p)$

it would prefer to choose rule B and assert d and f since it introduces less number of events than choosing A. However, if b is known to have occurred at time t_a, it would choose rule A since there is more supporting evidence.

BCI set

In order to incorporate the notion of backward persistence outlined above, we construct sets which are *chronologically ignorant backwards* (bci sets). Guided by the preference criteria listed in the previous section, we construct a bci set $S_{\Uparrow}$ for a causal theory Ψ. Hereafter we use $M_{\Downarrow}$ for the forward model and $S_{\Uparrow}$ for the backward set. Following is the procedure to construct $S_{\Uparrow}$. We shall see the case where inertial rules are part of causal theories, since it subsumes the case where inertial rules are not used.

BCI set with inertial rules

Here we manipulate seven sets, PRECONDITIONS, NEW-B-POTENTIALS, INITIATION, B-CLIPPED, B-POTENTIALS, PARTIAL-ANTECEDENTS and ANTECEDENTS. NEW-B-POTENTIALS, INITIATION, B-CLIPPED, and B-POTENTIALS roughly correspond respectively to NEW-POTENTIALS, NATURAL-DEATH, CLIPPED, and POTENTIALS in the forward model construction.

1. Let t_ω be a time point defined as the ltp of any term φ such that $\Theta \supset \Box\varphi$ is a boundary condition. t_ω can be regarded as the point where we start looking back into past events. For any φ whose ltp $\geq t_\omega$, we can define a time bounded set $S_{\Uparrow}/t_\omega \vdash \Box\varphi$.

   Initial B-POTENTIALS and ANTECEDENTS can be defined as an empty set, hence

   $\text{B-POTENTIALS}_{t_\omega} = \{\}$
   $\text{ANTECEDENTS}_{t_\omega} = \{\}$

We then augment $S_{\mathfrak{N}}/t$ to $S_{\mathfrak{N}}/t-1$ as follows.

2. PRECONDITIONS

PRECONDITIONS comprises of two subsets, BOUNDARY-CONDITIONS and INVOKED-CONDITIONS.

$$\text{BOUNDARY-CONDITIONS}_{t-1} = \{\Box\varphi : \Box\varphi \in \Psi, \; ltp(\varphi) = t-1\}$$

$$\begin{aligned}\text{INVOKED-CONDITIONS}_{t-1} = \{&\Box\varphi : \Phi \wedge \Theta \supset \Box\varphi_1,\\ &etp(\varphi_1) = t+1,\\ &\Box\varphi_2 \in \Phi \wedge \Box\varphi_2 \in \text{BOUNDARY-CONDITIONS}_{t-1},\\ &S_{\mathfrak{N}}/t \vdash \Box\varphi_1,\\ &\Box\varphi \in \Phi, ltp(\varphi) = t\}\\ \cup\\ \{&\Box\varphi : \Phi \wedge \Theta \supset \text{POTEN}(t+1, t_2, \mathbf{p}{-}p),\\ &\text{B-PROJECT}(t, \mathbf{p}{-}p, t, t_2, p),\\ &\Box\varphi_2 \in \Phi \wedge \Box\varphi_2 \in \text{BOUNDARY-CONDITIONS}_{t-1},\\ &S_{\mathfrak{N}}/t \vdash \Box(t+1, t+1, p),\\ &\Box\varphi \in \Phi, ltp(\varphi) = t\}\end{aligned}$$

etp and ltp stands for 'earliest time point' and 'latest time point' respectively.

The first set (BOUNDARY-CONDITIONS) consists of the boundary conditions which hold at $t-1$. The second set (INVOKED-CONDITIONS) consists of sets of conditions which are likely to be true given the presence of boundary conditions in BOUNDARY-CONDITIONS and the consequent of the causal rules and inertial rules being true in the time bounded set.

It is important to note that the subsets in INVOKED-CONDITIONS are sorted so as to fulfill the minimal cause criterion described in Section 4.4.2. The procedure for this sorting is as follows:

(a) Let m be the number of conditions which exists in both subsets of INVOKED-CONDITIONS and BOUNDARY-CONDITIONS. Sort INVOKED-CONDITIONS in the descending order of m. (*Preference for more supporting evidence.*)

(b) Let n be the number of conditions in each subset which are not members of BOUNDARY-CONDITIONS. In the sorted INVOKED-CONDITIONS above, within the subsets which have the same m, order them in the ascending order of n. (*Preference for minimal causes.*)

These two sorting procedures re-orders INVOKED-CONDITIONS in the order of preference for more supporting evidence and minimal causes. Choose the first subset g_1 in INVOKED-CONDITIONS and create

$$\text{PRECONDITIONS}_{t-1} = \text{BOUNDARY-CONDITIONS}_{t-1} \cup g_1.$$

3. NEW-B-POTENTIALS

$$\begin{aligned}
\text{NEW-B-POTENTIALS}_{t-1} = \{\langle t_1, t_2, \mathrm{p}{-}p\rangle : {} & \Box(t_3, t_4, p) \in \text{PRECONDITIONS}_{t-1}, \\
& \text{POTEN}(t_1, t_2, \mathrm{p}{-}p) \in \Psi, \\
& t_1 \le t_3, t_4 \le t_2\} \\
\cup \\
\{\langle t_1, t_2, \mathrm{p}{-}p\rangle : {} & \Box(t_3, t_4, p) \in \text{PRECONDITIONS}_{t-1}, \\
& \Phi \wedge \Theta \supset \text{POTEN}(t_1, t_2, \mathrm{p}{-}p) \in \Psi, \\
& t_1 \le t_3, t_4 \le t_2\} \\
\cup \\
\{\langle t_1, t_2, \mathrm{p}{-}p\rangle : {} & S_\eta/t \vdash \Box(t_3, t_4, p), \\
& \Phi \wedge \Theta \supset \text{POTEN}(t_1, t_2, \mathrm{p}{-}p) \in \Psi, \\
& t_1 \le t_3, t_4 \le t_2\} \\
\cup \\
\{\langle t_1, t_2, \mathrm{p}{-}p\rangle : {} & S_\eta/t \vdash \Box(t_3, t_4, p), \\
& \Phi \wedge \Theta \supset \text{POTEN}(t_1, t_2, \mathrm{p}{-}p) \in \Psi, \\
& t_1 \le t_3, t_4 \le t_2\}
\end{aligned}$$

NEW-B-POTENTIALS contains the potential history which was initiated by the events in PRECONDITIONS and those were asserted in the previous time

bounded set.

4. INITIATION

$$\text{INITIATION}_t = \{\Box(t, t_1, p) : \langle t, t_1, p\rangle \in \text{B-POTENTIALS}_t\}$$

INITIATION is a set of potential histories which were the cause of the backward persistence. The persistence ended due to its initiation, in other words, the time forward persistence initiated has been identified to be $t-1$ merely because it was their lower bound.

5. B-CLIPPED

$$\begin{aligned}\text{B-CLIPPED}_t = \{\Box(t, t_1, p) : {} & \langle t_1, t_2, p\rangle \in \text{B-POTENTIALS}_t,\\ & \text{B-PROJECT}(t-1, \text{p}{-}p, t_2, t_3, [\neg]p) \in \Psi,\\ & \Box(t-1, t_3, \neg[\neg]p) \notin \text{PRECONDITIONS}_t\}\\ \cup\\ \{\Box(t, t_1, p) : {} & \langle t_1, t_2, p_1\rangle \in \text{B-POTENTIALS}_t,\\ & \text{B-PROJECT}(t-1, \text{p}{-}p_1, t_2, t_3, p_1) \in \Psi,\\ & \Box(t-1, t_3, p_1) \in \text{PRECONDITIONS}_t,\\ & p \text{ and } p_1 \text{ are contradictory}\}\\ \cup\\ \{\Box(t, t_1, p) : {} & \langle t_1, t_2, p\rangle \in \text{B-POTENTIALS}_t,\\ & \text{B-PROJECT}(t-1, \text{p}{-}p, t_2, t_3, [\neg]p) \in \Psi,\\ & \Box(t-1, t_3, \neg[\neg]p) \in \text{ANTECEDENTS}_{t-1}\}\\ \cup\\ \{\Box(t, t_1, p) : {} & \langle t_1, t_2, p_1\rangle \in \text{B-POTENTIALS}_t,\\ & \text{B-PROJECT}(t-1, \text{p}{-}p_1, t_2, t_3, p_1) \in \Psi,\\ & \Box(t-1, t_3, p_1) \in \text{ANTECEDENTS}_{t-1},\\ & p \text{ and } p_1 \text{ are contradictory}\}\end{aligned}$$

B-CLIPPED is a set of events which are the remainder of the potential histories which ended at t. As was in CLIPPED in forward modelling, it means that if they were extended to $t-1$, they would have caused a contradiction.

6. B-POTENTIALS

$$\text{B-POTENTIALS}_{t-1} = \text{B-POTENTIALS}_t \cup \text{NEW-B-POTENTIALS}_{t-1} - \{\langle t_1, t_2, p\rangle : \Box(t, t_2, p) \in (\text{INITIATION}_t \cup \text{B-CLIPPED}_t)\}$$

B-POTENTIALS contains the potential histories which persist until $t-1$, and have potential to persist further back.

7. PARTIAL-ANTECEDENTS

$$\begin{aligned}\text{PARTIAL-ANTECEDENTS}_{t-1} = \{\Box\varphi : {} & \Phi \wedge \Theta \supset \Box(t, t_1, p) \in \Psi,\\ & S_{\text{fl}}/t \vdash \Box(t, t_1, p),\\ & \langle t, t_1, \mathrm{p}-p\rangle \notin \text{B-POTENTIALS}_{t-1},\\ & \Box\varphi \in \Phi,\\ & ltp(\varphi) = t-1,\\ & \Box\neg\varphi \notin \text{PRECONDITIONS}_{t-1},\\ & \Box\varphi \text{ is not contradictory with}\\ & \text{any term in PRECONDITIONS}_{t-1}\}\\ \cup\\ \{\Box\varphi : {} & \Phi \wedge \Theta \supset \text{POTEN}(t, t_1, \mathrm{p}-p),\\ & \Box\varphi \in \Phi,\\ & ltp(\varphi) = t-1,\\ & \Box\neg\varphi \notin \text{PRECONDITIONS}_{t-1},\\ & \Box\varphi \text{ is not contradictory with}\\ & \text{any term in PRECONDITIONS}_{t-1}\}\\ \cup\\ & \text{PRECONDITIONS}_{t-1}\end{aligned}$$

Condition $\langle t, t_1, \mathrm{p}-p\rangle \notin \text{B-POTENTIALS}_{t-1}$, guarantees to choose the potential history if any exists. This means causal rules are chosen only when there the proposition is not a part of a potential history.

8. ANTECEDENTS

$$\text{ANTECEDENTS}_{t-1} = \text{PARTIAL-ANTECEDENTS}_{t-1} \cup \{\Box(t-1, t_3, [\neg]p) : \langle t-1, t_3, \text{p}{-}p\rangle \in \text{B-POTENTIALS}_{t-1}, \text{B-PROJECT}(t_1, \text{p}{-}p, t-1, t_3, [\neg]p) \in \Psi\}$$

PARTIAL-ANTECEDENTS plus what you can conclude from the backward projection of potential histories.

$S_{\Psi}/t-1$ is obtained by making all wffs in ANTECEDENTS_{t-1} of the form $\Box(t,t,p)$ true and making all wffs of the form $\Box(t,t,\neg p)$ false.

The choice of alternative sets could be made by selecting an alternative set for the INVOKED-CONDITIONS during the construction of PRECONDITIONS.

Since the construction of partial sets proceeds backwards only, they are not reconstructed on the basis of occurrence of a boundary conditions on the process. One such example is where there is a supporting boundary condition for a causal rule in a construction of PARTIAL-ANTECEDENTS set. Given two causal rules

a. $\Box(t,t,p) \wedge \Box(t,t,a) \supset \Box(t+1,t+1,q)$

b. $\Box(t,t,p) \wedge \Box(t,t,b) \supset \Box(t+1,t+1,q)$

and $(5,5,q) \vdash S_{\Psi}/5$. There are two possible sets for $\text{PARTIAL-ANTECEDENTS}_4$, which are $\{\Box(4,4,p), \Box(4,4,a)\}$ and $\{\Box(4,4,p), \Box(4,4,b)\}$. These are treated as having equal priority. Now if there is a boundary condition at time 4, such that $\text{PRECONDITIONS}_4 = \{\Box(4,4,b)\}$. There is no selection criterion in the set construction which prefers, according to the higher plausibility, the occurrence of $\Box(4,4,b)$ in $\text{PARTIAL-ANTECEDENTS}_4$. This is dealt with in the construction of PRECONDITIONS where less occurrence of events is preferred among the selection of ANTECEDENTS. In this example, $\text{PRECONDITIONS}_4 = \{\Box(4,4,p),\Box(4,4,b)\}$ is preferred to $\text{PRECONDITIONS}_4 = \{\Box(4,4,p),\Box(4,4,a),\Box(4,4,b)\}$, providing the desired selection.

The set construction procedure described above has been implemented in SICStus Prolog (see Appendix C.2). Figure 4.3 illustrates the construction of S_{η} with inertial rules for Example 3.

It is important to note that, unlike the forward cmi model, the backward set is not necessarily unique.[9] This is obvious since there could be multiple causes for an event in the causal theory, where no preference can be assigned. In the forward formulation, the consistency condition restricted that only a single consequent can be awarded to the same antecedent.

4.5 Bidirectional sweeping over the interpolation range

This section discusses the issues surrounding backward persistence and introduces the notion of bidirectional persistence to treat interpolation problems.

4.5.1 Backward persistence

Whether an event should persist backwards has been an issue in the discussion about persistence. Kautz[86] did not address the possibility of applying persistence backwards to cater for SCP, but pointed out that forward persistence is not enough to solve the problem. Shanahan[89] claimed that persistence should only work forwards based on the observation that an event must have a cause that initiated it, and it is almost impossible to identify when that cause took place. This idea has the support of the notion of the 'autonomy of the past', which guarantees that what happens in the future does not affect the past. This is a reasonable assumption as long as we are talking about the nature of causation in a strictly physical sense. It is, however, not always the case when reasoning about the past—every so often we all try to make a reasonable reconstruction of the past.

We often think about the events in the future depending on the information available. When new information is added, we often change what we concluded previously, thus incurring the nonmonotonic nature of reasoning. The same can be said about the

[9] In the current implementation, it suggests one set at a time.

Causal theory Ψ:

1. $\Box$(t,t,glass-window) $\wedge$ $\Box$(t,t,vandalism) $\supset$ $\Box$(t+1,t+1,break-glass).
2. $\Box$(t,t,glass-window) $\wedge$ $\Box$(t,t,accident) $\supset$ $\Box$(t+1,t+1,break-glass).
3. $\Box$(t,t,glass-window) $\wedge$ $\Box$(t,t,gale-force-wind) $\supset$ $\Box$(t+1,t+1,break-glass).
4. $\Box$(t,t,break-glass) $\supset$ POTEN(t+1,∞,p-broken-glass).
5. $\Box$(t,t,broken-glass) $\wedge$ $\Box$(t,t,replace-glass) $\supset$ $\Box$(t+1,t+1,glass-window).
6. POTEN(1,∞,p-glass-window).
7. $\Box$(6,6,glass-window).
8. B-PROJECT(t_1,p-glass-window,t1,t,glass-window).
9. B-PROJECT(t_1,p-broken-glass,t1,t,broken-glass).

PRECONDITIONS$_6$ = {(6,6,glass-window)}
NEW-B-POTENTIALS$_6$ = {(1,6,p-glass-window)}
INITIATION$_6$ = {}
B-CLIPPED$_6$ = {}
B-POTENTIALS$_6$ = {(1,6,p-glass-window)}
PARTIAL-ANTECEDENTS$_6$ = {(6,6,glass-window)}
ANTECEDENTS$_6$ = {(6,6,glass-window)}

PRECONDITIONS$_5$ = {}
NEW-B-POTENTIALS$_6$ = {}
INITIATION$_5$ = {}
B-CLIPPED$_5$ = {}
B-POTENTIALS$_5$ = {(1,6,p-glass-window)}
PARTIAL-ANTECEDENTS$_5$ = {}
ANTECEDENTS$_5$ = {(5,5,glass-window)}

PRECONDITIONS$_4$ = {}
NEW-B-POTENTIALS$_4$ = {}
INITIATION$_4$ = {}
B-CLIPPED$_4$ = {}
B-POTENTIALS$_4$ = {(1,6,p-glass-window)}
PARTIAL-ANTECEDENTS$_4$ = {}
ANTECEDENTS$_4$ = {(4,4,glass-window)}

.....

PRECONDITIONS$_1$ = {}
NEW-B-POTENTIALS$_1$ = {}
INITIATION$_1$ = {}
B-CLIPPED$_1$ = {}
B-POTENTIALS$_1$ = {}
PARTIAL-ANTECEDENTS$_1$ = {(1,6,p-glass-window)}
ANTECEDENTS$_1$ = {(1,1,glass-window)}

Figure 4.3: A bci set for glass window problem: inertial theory.

past. We think about the past from the information provided to come up with the most plausible set of what happened in the past. This may be updated or changed by the addition of new information. In this sense, application of the notion of backward persistence is justified: if something persists into the future, there is no reason to dismiss the validity of persistence into the past. If conclusions derived from backward persistence are to be declared invalid, then so are those from forward persistence. Persistence can be seen as a ripple in the water created after a stone has been thrown in where it propagates in every direction. In this 'ripple model' of persistence, the stone (or the incidence of the ripple) is the observed event which has the property of persistence, and the ripple is the wave of persistence which propagates until it is obstructed by other events.

4.5.2 Bidirectional persistence

It is evident from the problems such as SCP that, in an interpolation problem, simply applying the preference condition of persistence to a single direction does not always result in plausible explanations of the past. The strategy here to deal with such problem is to construct both forward and backward sets for the same causal rules and set of boundary conditions. For convenience, we use the term *sweeping* to mean the construction of models and bci sets. It can also be said that due to the nature of the persistence measure, it is likely that the forward and backward sweeping results in the different temporal distribution of events. However, there are sets which are consistent in both directions, i.e., the forward model and backward set agree on the occurrence and temporal order of events.

Definition 13 (Consistency of cmi models and bci sets) *A bci set is* consistent *with the cmi model when for every sentence in the bci set, it is true in the cmi model.*

Definition 14 (Reconciliation set) *A bci set which is consistent with the cmi model is called a* reconciliation set.

Given a causal theory Ψ and its cmi model and a bci set, the following two cases can be considered.

1. If the cmi model and the bci set are consistent: the bci set is a reconciliation set.

2. If the cmi model and the bci set are inconsistent: the causal theory Ψ can be extended to a theory Ψ' so as to remove some inconsistency between the cmi model and the bci set.

The second case can be realised by adding a boundary condition consistent within the interpolation range to make the cmi model and the bci set come closer. This operation can be repeated with the aim of obtaining a reconciliation set.

Reconciliation algorithm

The algorithm to construct a reconciliation set is as follows. M_Ψ and S_Π represent, respectively, the cmi model and a bci set for a causal theory Ψ for a given interpolation range.

1. Starting from the earliest time point, identify the contradiction between the sets, i.e., contradicting events and their time range. For example, $M_\Psi \models (4, 4, a)$ and $S_\Pi \vdash (4, 4, \neg a)$ is a contradiction. Collect all such contradictions until the latest time point of the two sets.

2. For each contradiction, identify an event which is the cause of contradiction. This can be done by taking into account the causal rules or inertial rules responsible for the occurrence of contradictory events.

3. For a pair of contradictory events, say a and $\neg a$, where the time range of the contradiction $t_1 \leq t \leq t_2$, and the cause of contradiction p, a consistent set can be obtained by assuming the occurrence of p at t_p such that $t_1 - 1 \leq t_p \leq t_2 - 1$. Add $\Box(t_p, t_p, p)$ to the causal theory Ψ. Perform this operation for all such pairs, and construct M'_Ψ and S'_Π for the new causal theory Ψ'.

M'_Ψ and S'_Π for Ψ' constructed in this manner in practice provide a reconciliation set, though in general some new contradiction may have arisen. See Appendix C.3 for

the Prolog implementation of contradiction detection and event suggestion to resolve contradiction. In theory, there can be cases where the addition of new boundary conditions results in creating M'_{ψ} and S'_{η} with other inconsistent events. In an effort to resolve them the reconciliation algorithm can be repeated; this may result in adding a substantial number of boundary conditions which would prevail over causal rules in the theory obtained, dictating the occurrence of events.

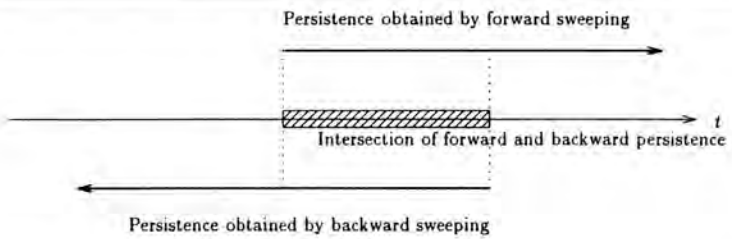


Figure 4.4: Capturing the intersection of the forward and backward sweeping.

4.5.3 Stolen car problem revisited

Provided with the logical framework for a chronologically ignorant set backwards in time, let us go back to SCP (Example 2). A causal theory for SCP can be formulated as follows:[10]

1. $\Box(t,t,park) \supset \text{POTEN}(t+1,\infty,\text{p-}parked)$.
2. $\Box(t,t,\neg parked) \supset \text{POTEN}(t+1,\infty,\text{p-}\neg parked)$.
3. $\Box(t,t,parked) \wedge \Box(t,t,stolen) \supset \Box(t+1,t+1,\neg parked)$.
4. $\Box(t,t,parked) \wedge \Box(t,t,towed\text{-}away) \supset \Box(t+1,t+1,\neg parked)$.
5. $\Box(1,1,park)$.

[10] In the causal theories for SCP and sample problems which appear later, projection rules are omitted for simplicity. We assume that a projection rule exists for each potential rule provided.

6. $\Box(10,10.\neg parked)$.

By sweeping forward, we can construct the forward cmi set

$$M_{\Downarrow} = \{(1,park),(2,parked),(3,parked),(4,parked),(5,parked),(6,parked),(7,parked),(8,parked),(9,parked),(10.\neg parked)\}$$

Similarly, a bci set is

$$S_{\Uparrow} = \{(1,park),(2,parked),(2,stolen),(3,\neg parked),(4,\neg parked),(5,\neg parked),(6,\neg parked),(7,\neg parked),(8,\neg parked),(9,\neg parked),(10,\neg parked)\}$$

where $(2,stolen)$ may be replaced equivocally with $(2,towed\text{-}away)$.

The model and the set clearly illustrate the difference between forward and backward persistence. The forward model stipulates that the car was parked until immediately before it was gone, while the backward set shows that the car was stolen just after it was parked. An interesting point is that the forward model cannot identify the cause for the car being $\neg parked$. Focussing on the $parked/\neg parked$ events, the forward model and the backward set agree on the occurrence of events $\{(1,park),(2,parked),(10.\neg parked)\}$. Between $t = 3$ to 9, the two contradict over this proposition.

What we can conclude from the difference is that this portion of the history cannot be determined. According to $S_{\Uparrow}$, the cause responsible for the occurrence of $\neg parked$ is *stolen*. Since there is no way to decide when $\neg parked$ event initiated, the event *stolen*, which was introduced as a hypothesis in $S_{\Uparrow}$, cannot be assigned with the definite time point. The possibilities are, however, that it happened during the range of $t = 2$ and $t = 9$. Asserting this back to the causal theory, using for instance $\Box(5,stolen)$, the histories given by the forward and backward sweeping will be exactly the same,

$$M = \{(1,park),(2,parked),(3,parked),(4,parked),(5,parked),(5,stolen),(6,\neg parked),(7,\neg parked),(8,\neg parked),(9,\neg parked),(10,\neg parked)\}$$

What we have done here is to examine differences in the intersection of forward and backward sets (Figure 4.4). By identifying what events are missing, which produced the difference between the two sets, we can add them in the causal theory to obtain several sets, any of which may account for the plausible interpolation of events.

4.6 Application of bidirectional sweeping to sample problems

Sandewall compiled and analysed several test scenarios for a variety of prediction and reconstruction tasks [Sandewall 92b, Sandewall 92a]. [11] Below are the two problems from among those which Sandewall analysed to be unsolvable by the logic CI. We shall see here how these problems are coped with bidirectional sweeping.

Example 4 (Stanford Murder Mystery (SMM)) *Initially there is a live turkey. At a certain time point, the firing event takes place and after that the turkey is dead. The problem is to identify occurrences of events to complete this scenario. (Originally introduced by [Baker 89].)*

This is a modified version of the Yale Shooting Problem, and the main feature is that this involves a reconstruction task. This is an interpolation task since the scenario is bounded at both ends by the turkey being alive and it being dead. We can adapt the causal theory used for YSP to formulate the causal theory for this problem.

1. $\Box(t,t,turkey(alive)) \wedge \Box(t,t,loaded(gun)) \wedge \Box(t,t,fire(gun)) \wedge \Diamond(t,t,bullet(real)) \supset \text{POTEN}(t+1,\infty,\text{p-}turkey(dead))$.
2. $\Box(t,t,load(gun)) \wedge \Box(t,t,\neg loaded(gun)) \supset \text{POTEN}(t+1,\infty,\text{p-}loaded(gun))$.
3. $\text{POTEN}(1,\infty,\text{p-}turkey(alive))$.
4. $\Box(5,5,fire(gun))$.
5. $\Box(6,6,turkey(dead))$.

[11] Sandewall characterises these problems as *chronical completion tasks* which is to identify the set of all occurrences of events in a scenario where all actions are described in well-defined, sequential order.

6. $\Box(6,6,\neg loaded(gun))$.

By sweeping forwards, we can obtain the following cmi model.

$M_{\Downarrow} = \{ (1,turkey(alive)),(2,turkey(alive)),(3,turkey(alive)),$
$(4,turkey(alive)),(5,turkey(alive)),(5,fire(gun)),$
$(6,turkey(dead))\}$.

This model suggests that the turkey was alive until immediately before the gun was fired and is dead immediately after for no particular reason, since nothing is said about the status of the gun being loaded or unloaded.

By sweeping backwards, the following bci set can be obtained.

$S_{\Uparrow} = \{ (6,turkey(dead)),(5,fire(gun)),(5,turkey(alive)),$
$(5,loaded(gun)),(4,turkey(alive)),(4,loaded(gun)),$
$(3,turkey(alive)),(3,loaded(gun)),(2,turkey(alive)),$
$(2,loaded(gun)),(1,turkey(alive)),(1,load(gun)),$
$(1,\neg loaded(gun))\}$.

This set suggests that the gun was loaded at time 1, being kept loaded until the fatal firing event.

In this case, there is no explicit contradiction between the cmi model and the bci set since nothing is said about the gun in cmi model to contradict the bci set. However, bci set provides more information about the gun if required.

An interesting case in this scenario is when the gun was initially not loaded, and the gun remains unloaded unless a load event occurs.

6. POTEN$(1,\infty,p\text{-}\neg loaded(gun))$.

In this case, the bci set is unaffected, but the cmi model is

$M_{\Downarrow} = \{ (1,turkey(alive)),(1,\neg loaded(gun)),(2,turkey(alive)),$

$(2,\neg loaded(gun)),(3,turkey(alive)),(3,\neg loaded(gun)),$
$(4,turkey(alive)),(4,\neg loaded(gun)),(5,turkey(alive)),$
$(5,fire(gun)),(5,\neg loaded(gun)),(6,turkey(dead)),$
$(6,\neg loaded(gun)),\}$.

There are contradictions of loaded(gun) and ¬loaded(gun) for the time range of 2 to 5. In order to resolve this contradiction, a load(gun) event must occur between 2 and 4.

Sandewall suggests the solution to this problem using his framework that the gun was *initially* loaded under both cases above. This is clearly undermining the possibility of the gun being loaded just before the firing event took place, which is covered by bidirectional sweeping.

Example 5 (Ticketed car scenario (TCS)) *The single yellow line region of a street typically is a 'no parking' zone during daytime but parking is allowed overnight (between 5pm and 8am). One evening, you left the car, initially unticketed, for two whole nights and days to find it ticketed in the third evening.*[12] *The problem is to show when ticketing event took place.*

This is a modified version of the Stolen Car Problem but with conditions to the ticketing event to take place, i.e., it can only happen during daytime. The causal theory for this problem can be formulated as below.

1. $\Box(t,t,park(car)) \supset \text{POTEN}(t+1,\infty,\text{p-}parked(car))$.
2. $\Box(t,t,ticket(car)) \supset \text{POTEN}(t+1,\infty,\text{p-}ticketed(car))$.
3. $\Box(t,t,parked(car)) \wedge \Box(t,t,ticket(car)) \wedge \Box(t,t,\neg ticketed(car)) \wedge \Box(t,t,time(day)) \supset \Box(t+1,t+1,ticketed(car))$.
4. $\Box(1,1,park(car))$.
5. $\Box(1,3,time(night))$.

[12] In the scenario by Sandewall, daytime parking was allowed but no overnight parking.

6. $\Box(4,6,time(day))$.

7. $\Box(7,9,time(night))$.

8. $\Box(10,12,time(day))$.

9. $\Box(13,15,time(night))$.

10. POTEN$(1,\infty,\neg ticketed(car))$,

11. $\Box(14,14,ticketed(car))$.

The cmi model obtained by forward sweeping and the bci set constructed by backward sweeping are shown respectively in Tables 4.1 and 4.2.

Table 4.1: The cmi model for TCS.

time			
1	*time(night)*	*park(car)*	$\neg$*ticketed(car)*
2	↓	*parked(car)*	↓
3	↓	↓	↓
4	*time(day)*	↓	↓
5	↓	↓	↓
6	↓	↓	↓
7	*time(night)*	↓	↓
8	↓	↓	↓
9	↓	↓	↓
10	*time(day)*	↓	↓
11	↓	↓	↓
12	↓	↓	↓
13	*time(night)*	↓	↓
14	↓	↓	*ticketed(car)*

By comparing the cmi model and the bci set, it is apparent that the two contradict between time points 5 and 13. Therefore the event *ticket(car)* should have occurred between time points 4 and 13. However, since there is is a restriction as to when the ticketing can take place, i.e., it can only occur during daytime, it should have occurred either between 4 and 6, or between 10 and 12.

Table 4.2: The bci set for TCS.

time			
1	⊤	*park(car)*	¬*ticketed(car)*
2	⊤	⊤	⊤
3	*time(night)*	⊤	⊤
4	⊤	⊤	⊤
5	⊤	⊤	⊤
6	*time(day)*	⊤	⊤
7	⊤	⊤	⊤
8	⊤	⊤	⊤
9	*time(night)*	⊤	⊤
10	⊤	⊤	⊤
11	⊤	⊤	⊤
12	*time(day)*	⊤	⊤
13	⊤	⊤	⊤
14	*time(night)*	*parked(car)*	*ticketed(car)*

4.7 On abduction and backward reasoning

This section discusses issues concerning the use of abduction and its applications for backward reasoning.

4.7.1 Abductive reasoning

Inherent complexity of abduction

The main problem with abduction using causal rules is that Shoham's form of causal rules are designed to suit the forward progression of time. The syntactic restrictions and the consistency condition limit the branching factor forward, while increasing the complexity backwards (Figure 4.5). As the diagram shows, it typically forms a tree which has leaves in the past and a root in the future. While the events which are derived from a set of conditions are always consistent owing to the consistency condition, we can have inconsistent events as the possible candidates for the cause of an event. This can be seen in the following simple example: a causal theory which comprises two causal rules $\Box(t,t,p) \wedge \Diamond(t,t,\neg d) \supset \Box(t+1,t+1,q)$ and $\Box(t,t,\neg p) \wedge \Diamond(t,t,\neg d) \supset$

$\Box(t+1, t+1, q)$ is sound and consistent. If there is $\Box(2, 2, p)$ it will conclude $\Box(3, 3, q)$. However, $\Box(3, 3, q)$ would have, as its cause, two inconsistent candidates $\Box(2, 2, p)$ and $\Box(2, 2, \neg p)$.

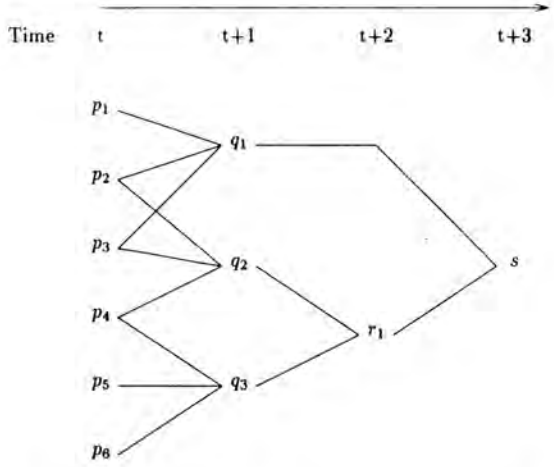


Figure 4.5: An example of causal links of events in temporal order in a sound and consistent causal theory. If we follow the time line forward, the tree is converging, while following backwards increases the branches.

Abduction of causal rules

There are two kinds of abduction which can be performed in the framework of causal theories. One is reasoning about the events which precede an event given a causal theory. This is abduction over causal rules. The other is to construct a theory itself from a set of events, which is the abduction of a causal theory. The former is applying causal rules backwards, from effect to cause, to identify the events that caused certain events. For instance, if there is a causal rule which states, "smoking causes lung cancer", and given an actual case of lung cancer, one may conclude that smoking had

happened sometime before the occurrence of cancer.[13] The latter is hypothesising and testing a theory which would explain the connection between the events which are likely to have some kind of causal relation. It is analogous to the formation of a scientific theory from observations of physical phenomena, for instance the discovery of radioactivity by Antoine-Henri Becquerel. This type of abduction is discussed in Chapter 7.

There have been other approaches in advocating abduction to reason about the causes for an event and the construction of a theory. An area in which abduction has been well investigated recently is in the diagnosis task. Konolige[92] combines default and causal reasoning using a default causal net (DCN) to predict the outcome of a set of events, and explain why a malfunction occurred and which component is responsible. Ng and Mooney[92] conducted empirical investigations for the effectiveness and efficiency of abductive reasoning in plan recognition and diagnosis.

The significance of performing abductive reasoning in the framework of CI is to provide a facility to model the past from causal theories, which is a powerful tool for modelling the future. In this framework, we do not need to provide rules in 'if effect then cause' format which is counter-intuitive.

4.7.2 Sweeping

Boddy *et al.*[92] take a similar approach to sweeping in both directions. The language they developed is based on Dean's *Time Map Manager*(TMM)[Dean & McDermott 87]. It allows the predicates $\text{Persists}_F(T_1, T_2, P)$ to describe that a proposition P persists *forward* over times T_1 and T_2 and, similarly, $\text{Persists}_B(T_1, T_2, P)$ to persist *backwards*. This persistence knowledge is used to provide maximal persistence in both directions, which forms the basis for backward set. However, the introduction of backward persistence rules specifically describe that a certain event persists backward. This approach is analogous to providing 'if effect then cause' form of inverted implications as used in MYCIN but which is counter-intuitive and logically incorrect in this case.

[13] More precisely, smoking can be considered as a valid candidate for the cause of lung cancer.

In my approach, knowledge about persistence is always forward. The backward projection takes care of the application of the inertial rule backwards, which is analogous to the abductive use of causal rules. In this way, I avoid the assertion of knowledge for backward causation, which I consider counter-intuitive. In this respect, work by Morgenstern and Stein is closer to my approach [Morgenstern & Stein 88] but they do not address backward persistence as preferred criterion and treat all backward projection as equally acceptable.

4.8 Expressiveness vs complexity

Logic CI is well designed to provide a clear logical framework for reasoning about the future. In contrast, when applied backwards in time, this advantage may prove to be a difficulty. Since the consistency condition only applies forward in time, the branching factor into the past is naturally greater. This makes it difficult to control the choice of an appropriate premise to be abduced. However, the other features of the logic CI, such as the representation of basic causal rules and inertial rules, provide desirable expressiveness and clarity in dealing with causal knowledge.

In his book, Shoham states [Shoham 88, p.168]:

> Indeed, there is no "unique explanation" theorem that corresponds to the "unique cmi model" theorem, and correspondingly no easy algorithm to compute possible explanations that is analogous to the $O(nlogn)$ prediction algorithm. Why then should we stick with causal theories? Why not organize the knowledge differently so that, for example, prediction might become harder, but planning easier? [14]

The justification he emphasises is that the prediction is necessary in order to identify *what* to plan for. In other words, by anticipating the events in the future, the motivation is set for planning, either to achieve alternative outcomes, or to avoid such events occurring. The framework which deals with past events in the language of logic CI

[14] Shoham uses "planning" as a representative task of reasoning about preconditions (causes) as opposed to "prediction" which is reasoning about conclusions (effects).

provides a strong support for this position. We can conduct plausible reasoning about the past while enjoying a well-founded mechanism for prediction.

In most cases, reasoning about the past is actually reasoning about events that happened between two or more time points. Not only the starting point of reasoning, the present, but the earlier initial conditions are usually known. We reason about what has happened to a car which is wrecked because we know that the car was once in shape; we think about the past to find plausible ideas about what had happened that made the car as it is at the present. We called this an interpolation task. We suggested a way to deal with this, which is to sweep through the events in both directions, forward and backward, and find the place where the two sequences of events become inconsistent. From that point we can abduce events over causal rules in order to find the cause for the digression.

The major problem in abduction of causal theories is its computational complexity. While there is no obvious way to reduce the complexity syntactically, in actual applications it is possible to prepare a range of heuristics as to what can and cannot be seen as the cause for a particular event. For instance, if we provide knowledge about classes of events we might construct an abduction rule that an event of a class A cannot be a cause of an event of a class B. This should reduce the search space to a considerable extent. In the abduction-based diagnosis task described in Konolige[92], a (partial) fault model is prepared to guide the search for the most likely explanation of a fault. This provides the basis for the preference measure that the minimal number of primitive causes is the ideal explanation.

There might be an objection to applying persistence to temporal problems solving that it does not always provide plausible solutions. Kautz comments about the inadequacy of applying persistence to solve prediction and reconstruction problems:

> The inadequacy is ontological: we can't handle persistence properly until we have a richer theory of causation. The purely temporal solution often works because the flow of time reflects the order of physical causation. When the full story of causation is told, we then require an efficient

> algorithm for performing the necessary deductions, such as Hanks's, and a clear model theory such as that provided by generalized circumscription to explain and justify the whole process. [Kautz 86, p.405]

This statement nicely puts forward the necessity to devise a sound framework to perform reasoning under incomplete causal knowledge. The treatment of backward persistence and the application of bidirectional sweeping to interpolation problems described in this chapter is an attempt to support this cause.

4.9 Summary

In summary, this chapter has described how to deal with the reconstruction problem and formulated necessary mechanisms to perform this task in the framework of Shoham's causal theory. The major points are:

- The basic strategy is abduction over causal rules with preference for sets in which events potentially persist backwards in time.
- A framework for abduction to be performed for language CI preferring backward persistence is suggested and applied to reconstruction tasks. In this way, the same causal theory can be used for both prediction and reconstruction.
- For interpolation problems, a plausible explanation can be obtained by comparing the results of forward and backward sweeping.
- Although abduction over causal rules is itself computationally expensive, it can partially be controlled by preference for backward persistence.

This marks the end of the formulation of theoretical framework for causal reasoning. From the next chapter, I discuss the application of causal reasoning in design problem solving.

Chapter 5

Role of causal reasoning in design

In Chapters 2, 3, and 4 I have investigated the formalisms of causal reasoning and devised a framework of temporal logic to obtain the most likely prediction and reconstruction based on chronological minimisation. In this chapter, I consider how such a framework can be applied to solving design problems.

The major field of expert systems development to which temporal reasoning has been applied so far is planning and scheduling. Planning is naturally a temporal reasoning task since its purpose is to generate sequences of actions in order to meet the required task. This involves the notion of change, and requires some mechanism to manage temporal sequences. The general view about the difference between a planning task and a design task is that the latter concentrates on the configuration of components under certain constraints (See for instance [Hayes-Roth *et al* 83]). For instance, design of a table does not require the notion of change, but constraints on size, mass, strength, stability, etc. However, design problems are not just the problem of satisfying physical constraints. Clancey[85] places planning as a subproblem of design, together with configuration problems. From a designer's point of view, this is still not sufficient. In addition to these two tasks, a design problem should have as its subproblems modification, verification and simulation. These are all inter-related to solve even reasonably small design problems. There is no single representation of artefacts which caters for all these problems. Some of them are analysed as static knowledge while others as

dynamic. By this classification, the importance of representing temporal and dynamic information in design has not been emphasised as much as it should have been.

The following chapters discuss the role of causal reasoning in design, and how that can be supported by temporal logic described in the previous chapters. The aspect of design focused on here is that involving reasoning about time and change, which has not been given much attention until recently. Let us first begin by summarising the design theories and formal methods in design.

5.1 Design theory and methodology

Design is a discipline which is well appreciated as being one of the intelligent activities of humans and studied by engineers, architects and even philosophers. However, there is still a large gap between what is called 'design theory' on one hand and 'design methodology' on the other. A student of any engineering discipline studies various de sign methods of his/her specialised field: an electrical engineering student would design a flip flop circuit by drawing circuit diagrams and state diagrams; a mechanical engi neering student would design an engine by studying combustion, dynamics, materials, etc.; a chemical engineering student would design a processing plant by drawing flow diagrams, studying chemical reactions and a large amount of other related materials. These are guided by design methods which are usually well established through the history of creating artefacts in each particular field.

On the other hand, we may well think that there should be more to design than following instructions in the textbooks. Yoshikawa[80] proposed the General Design Theory (GDT), which advocates the idea that there is a framework in which various design activities can be modelled. GDT suggests that design is basically a configuration of necessary components which fulfills the design specification. This view of design can be considered to be a search into an appropriate combination of components or allocation of resources (See for example [Simon 69]). Although it is yet difficult to imagine such universal description of the character of design problem solving as GDT in real terms, the significance of GDT is in its attempt to formulate a general design

strategy which might be applied across various domains of design.

What makes design a complex task is not only the vastness of the search space in practical problems but also that it is inherently ill-structured [Simon 73]. It cannot be well formulated as solving a logic puzzle. The specification is not always well-defined, and it requires modelling and simulation to actually find out what would happen in reality. Minor points are left out to capture the overall picture of the project. It is almost impossible to create a single model of the project, and hence the problem must be seen from various aspects such as structure, function, appearance, etc. To further complicate the matter, integrating these aspects increases the complexity of the problem. The source of such complexity is that when a designer sets off designing, typically s/he does not have a clear picture of the problem nor the goal—these tend to become clearer as s/he actually becomes engaged in the design process. The analyses of design methodology might provide a better view of the design process at the earlier stages.

5.1.1 Characteristics of design problems

Among the characteristics of design tasks Lawson [Lawson 90, pp.88-93] suggests, the following points are most relevant when building a computer assisted design system.

Design problems cannot be comprehensively stated. This is one of the elements in design which makes it an ill structured problem. It is almost impossible to have a complete specification to start with and even then the specification can be inconsistent. The domain knowledge is not always complete and designers tend to omit minor details. This requires a facility to deal with frequent alterations in the specification as well as domain knowledge, and the use of declarative language is an advantage due to its modularity.

Design problems tend to be organised hierarchically. There are various levels of abstraction in the design problem. Also the view of design varies depending on what aspect of design is focused. Lawson states that there is no abstract logical way to select

the appropriate level of problems, but suggests that tackling it from a higher level is reasonable. This agrees with the position of the preliminary stage of design which is discussed in the later section.

There are an inexhaustible number of different solutions. Whether we want to obtain all possible solutions is another problem. What we typically require is one plausible solution, and a facility to get an alternative solution once the previous one becomes obsolete.

The design process involves finding as well as solving problems. This also constitutes the design problem as ill-structured problem. New problems arise in the course of an attempt to solve the current problem, and these contribute additional knowledge and assumptions about the problem. One problem may trigger a chain of problems and incrementally assert various constraints to the problem. The use of logic-based declarative languages to represent problems is an advantage under such conditions.

Design inevitably involves subjective value judgement. This provides a strong case against totally automated design systems. Unless the domain is well-defined (which is not usually the case as has been discussed) and the preference criterion to reflect the value is well-established, a design system requires interaction with the designer.

5.1.2 Design vs configuration

Some classify configuration and design in terms of the size of the search space. According to [Lawson 90], the former has smaller search space than the latter. In real-life problems, this does not seem to be the case. For instance, one typical configuration problem is that of the turbine generator plant for power stations which is highly complex and requires careful optimisation [MacCallum *et al* 92]. Such problems can be just as complex as any design problems.

Configuration is a task where given a set of components, we have to establish the connections between them in such manner that fulfills the specification. Perhaps the main difference between configuration and design is that the set of components and the specification are often well-defined and the main problem is the complexity of search. Design, on the other hand, does not always have a complete set of components and the specification is often ill-defined.

It should be noted that design and configuration are not separate problems. Various stages of design require configuration to shape up and clarify the problem and enhance the design process.

5.1.3 Design knowledge

Since design involves different kinds of knowledge which are not always compatible in terms of representation, it is inevitable that we need to represent them in different languages. It is useful to classify design knowledge according to how it is used and at what stage of the design process. An example of such a taxonomy of knowledge is by Dym[92] (Table 5.1) in the engineering domain.

Among the design knowledge as classified in Table 5.1, there has been less emphasis on knowledge concerned with time. Management of temporal constraints has been of less interest in the field of design than it has been in planning tasks.

Temporal knowledge (or simply "time") only appears explicitly as one of the fundamental types of knowledge, but other knowledge requires the notion of temporal ordering in implicit forms such as causality. For instance, phenomenological models which include various relations among physical parameters have causal relations in use, and so do analytical models. Many of the heuristics in the experimental knowledge are likely to involve causality. This invokes the importance of incorporating causal descriptions in the process of formalising design.

Table 5.1: A taxonomy of design knowledge. (from [Dym *et al* 92, p.4])

Engineering Domain Knowledge

- **Fundamental knowledge.** This is *first principles* knowledge about the relevant fundamental physical laws *and* the spatial and temporal references for their expression.
  - **Physical principles.** These are the basic physical laws underlining the domain in which a particular engineering problem is being solved.
  - **Geometry.** Geometrical knowledge includes descriptions of the shapes and forms of objects and the spatial relation of system components and other objects to each other.
  - **Time.** This knowledge is about ordering events and actions.
- **Phenomenological models.** These are models often based on experimental results or on compiled high-level extrapolations of the fundamental laws, e.g., pressure-volume relations, and generalized force-deflection relations.
- **Analytical models.** These are formulas, both exact and approximate, used to model specific "cases" or subsets of the more general first principles and phenomenological models, e.g., case-specific models for beams, columns, plates, and shells.
- **Numerical models.** These are the mathematical formulations of the computational versions of the analytical formulas just described, e.g., the matrix representations of variational statements that underlie finite element codes.
- **Various forms of system-level decompositions.** These decompositions form the "backbone" of the design approach from which other design decisions are investigated.
- **Component descriptions and synthesis knowledge.** The design of a complex system often involves selecting existing components and configuring them to meet certain objectives, as in the design and assembly of computer systems. The identification and representation of the attributes of these components is important engineering knowledge, as is the relationship of individual components to assemblies of components, i.e., parts-to-whole hierarchies.
- **Experimental knowledge.** This knowledge can take the (familiar) form of heuristics, but it also includes the codes and standards which serve as design objectives for many engineering artifacts.
  - **Heuristics.** Heuristics are typically written as rules because they represent those "rules of thumb" that arise out of experience.
  - **Codes, standards and constraints.** A complete engineering design is expected, based on our experience, to meet certain specifications or confirm to various code of behavior.

5.2 Causal reasoning in design

The aspect of design on which we focus here is that of a dynamic nature. That is, the description and analysis of an artefact whose feature is its performance over a certain duration of time, or over certain changes in the conditions of its surroundings. Such performance is seen as the *behaviour* of the artefact. What is advocated here is a *behaviour-oriented approach* to design, where the purpose of design is to obtain the desired behaviour of the artefact and the surroundings in which it operates, and to provide specification about how the artefact should behave rather than how it should be shaped.

Since behaviour is a sequence of how an artefact performs over a certain (finite) period of time, this can best be captured in term of temporal reasoning. Furthermore, behaviours are seen as the result of various causal connections among facts and events, and this fits the framework of causal reasoning. This section discusses what process of design can be seen as taking a "behaviourist" stance and what features of design involves the representation of the behaviour of an artefact.

5.2.1 Behavioural understanding in the preliminary stage of design

The general view of design as generation-and-test cycle can be applied to the various stages of the design process. In general terms, there are preliminary stages, where the design specification is evaluated and the overall conceptual model of design is constructed, and the later stage where the conceptual model becomes more concrete. In the preliminary stage, the focus is to capture the behavioural understanding of the device and the physical system in which it operates [Finger & Dixon 89]. The product of this stage is hence the rough sketch of design, pointing to its plausibility in the conceptual form. Some of the main activities in the preliminary stage of design are listed below [Nakata 92b]:

- Determine whether the design is achievable in principle.
- Reduce the search space by eliminating inappropriate partial solutions.

- Suggest "plausible" configurations for the specification.
- Reduce the number of "expensive" detailed (e.g. numerical) analyses.
- Help the designer to understand the problem and identify the problem area.

Another characteristic of the preliminary design is that despite its role as a mock-up process in the overall design, there are occasional feedbacks to and from the later stage of design. Every time relatively major modification has been made to the design, designers run a simple evaluation of the new design by means of preliminary analysis. Figure 5.1 illustrates the position of preliminary analysis in the overall cycle, where we can see that the preliminary analysis in not once-through stage in design but an essential subprocess. There has been less effort in supporting the preliminary stage of design as has been pointed out by Finn *et al*[92].

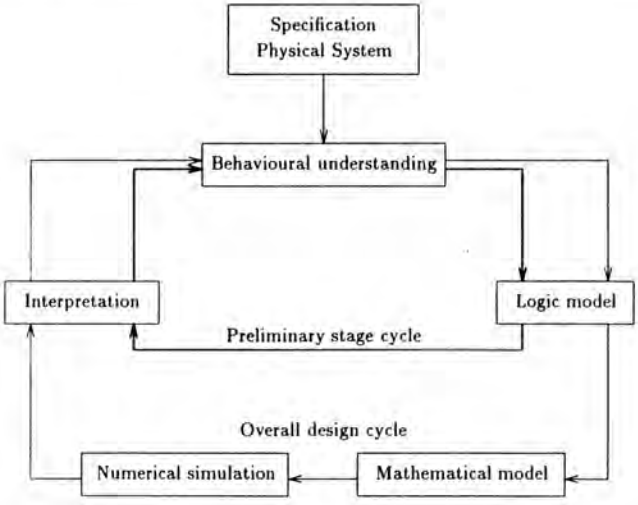


Figure 5.1: Position of preliminary stage in the design process(adapted from [Finn *et al* 92, p.580]).

From what we have seen, what we require in the preliminary stage of design is a facility to represent and analyse the behaviour of a device. A simplified version of this analysis is what human designers do even before starting to write down complicated mathematical equations. If you imagine yourself designing a device that uses the power of the wind to turn a generator, you probably would not start jotting down the equations to calculate the torque of the shaft, or the energy loss as the result of converting kinetic energy to electrical energy. A good starting point is probably drawing a rough sketch of what it would look like, and how things work over a certain period of time. This task, together with the mental simulation of the operation of the device, is dealt with using your abstract knowledge about causal relations among components and agents such as a gear and the wind. One way to describe what is occurring here is causal reasoning about 1) what is necessary in order to obtain desired effects, and 2) what should be expected to happen according to the current design. In other words, a simple form of means-ends analysis and plausibility evaluation is being conducted by using causal models.

5.2.2 Behaviour as causal model

Then how can behaviours be captured? The behaviour of a device in our terms is essentially how it performs under given conditions. We can perceive such performance of the physical object as a sequence of events occurring with the device. When we model the behaviour of the device, it is not adequate to simply examine how the device performs. Simon[69] suggests that an artificial system requires consideration of "inner" and "outer" environments. The "inner" environment is the device itself and its internal structures, while the "outer" environment is the surroundings in which the device is placed. An important point to consider is that there are interactions between both environments. A change in the "inner" environment may affect the condition of the "outer" environment, and vice versa. There obviously are some causal relations between these environments. The point is, we need to capture the behaviour of the whole system, i.e., both "inner" and "outer" environments and their interactions, when creating causal models.

As described in the previous section, design can be captured as an abductive process of deriving a structure from the specification. If we see the specification as a description of how the device should behave, the design task turns out to be the construction of a causal model in which the device behaves as intended, with possible interaction with other agents and the environment.

In terms of the classification of the tasks involving causal reasoning as we have seen in the previous chapters we can see the correspondence of the tasks in the following manner. The prediction task is the simulation of the behaviour of the device under certain conditions. The reconstruction task is analysing what caused certain events in the behavioural model. The actual design process is in fact an interpolation problem of identifying what should have happened between certain time points, where boundary conditions are set. This view is discussed further in the next chapter.

5.3 Formalising design

GDT was an attempt to create a formal model of what design is all about. It was an attempt to analyse the human activity of design and model it in a formal way in order to establish a formal method which applies to all sorts of design. It is now conceived that this is indeed a very difficult task, but at the same time an important task in building a computer model of design and computer support of design.

Here I present an overview of the abduction model of design and the general description of design process which is discussed in detail in the following chapters.

5.3.1 Logical description of design

One of the early approach to formalising design was that of design as a deductive process. This view was best described as "form follows function". If we have a strong theory about the design process then it follows that a specification will, quite naturally, deduce a design. However, as we have examined, design is not a well-structured task. If I borrow March's words, "when we are designing we do not have a clear picture of what the design is; if we did, our task is needless." [March 76, p.21]. Contrary to the

deduction model of design, a design is not what we obtain through deduction, but is the thing about which we make deductions. If we have a design, then we can deduce its functionality, features and validity. This leads to the idea that design is a reverse process of deduction. This process is what C. S. Peirce named *abduction*. Abduction is not a logically sound process, but can be seen in everyday human reasoning.

5.3.2 Abduction model of design

One of the systems which incorporated this abduction model of design is RESIDUE [Finger & Genesereth 85]. In this approach, the solution to a design problem is to find a *residue* for a design goal. More precisely, given the initial world model W and the design goal G, the solution is the *residue* R such that

$$W \cup R \vdash G \tag{5.1}$$

where $W \cup R$ is consistent and R can be made true. RESIDUE then applies a deductive procedure to derive R as the design solution, but it is essentially an abductive process since R is being derived as the realiser of G.

The design model I introduce is focused on the behavioural aspect of a device. The design goal is to provide a design that realises the desired behaviour in a designated environment. In this formulation, there are four elements involved for a device to exhibit its desired behaviour. There is an *environment* (E)—the world in which the device is meant to be placed and operate—which is governed and represented by a set of physical laws and unchangeable facts. There is also a set of isolated events, which we call a *scenario* (S), and this can be considered to be the events known or expected to occur independently from physical laws or the operation of the device. Scenarios provide, in the same environment, various situations under which the device is operated. It corresponds to a number of possible cases which need to be considered, depending on various assumptions. *Design* (D) is the knowledge about functions and the configuration of the device which would initiate a sequence of events which would not have occurred without its presence. These elements together form a *system* in which the device operates and produce a sequence of events called *behaviour* (or *predicted* behaviour) (B).

In the design task there is an additional element, the *behavioural specification* (BS). Since design activity is motivated to produce a desired behaviour of a system, there should be a specification for such behaviour. The behavioural specification of a device is the sequence of events and states which are desirable and necessary to its performance. Typically, a variety of behaviours is specified corresponding to different scenarios. All of these behavioural specifications require to be satisfied.

Given these elements, a successful design task can formally be described as follows. The behaviour of the system can be derived from the environment, a scenario and an artefact (a design or a partial design), i.e.,

$$E, S, D \models B \tag{5.2}$$

and the aim of design activity is to construct a design D which would result in the desired behaviour, $B = BS$. E and S are specified by the domain and setup of the design problem. In this abductive model of the design process, the analogy with the conventional iterative view of design method can be seen as follows; the evaluation process is the comparison of B and BS, the feedback is the abductive reasoning about D, which leads to the modification of design. Note that typically reasoning process to obtain B is nonmonotonic. It is easy to see that adding or removing an assumption in S or D may change the behaviour of the system. Additionally, since S and B consist of temporal sequences of events, the construction of the model B involves a form of temporal reasoning. The abductive process in design is discussed later in more detail in Section 7.3.

5.4 Overview of design phases

There are three major phases in a design level. These are simulation, verification and modification, and a design process is an iteration of these phases which terminates when the specification is achieved in the verification phase. In the proposed behaviour-oriented design process, each of these phases is constructed in the following way.

Simulation. The simulation of the behaviour of the system is given as the result of the prediction task. The prediction is based on the model constructed from E (environment), S (scenario) and D (design), which is obtained as the predicted behaviour (B). The language used for representation and reasoning is extended CI which was described extensively in Chapters 3 and 4.

Verification. Once the prediction of the behaviour is obtained through simulation, the next step is to verify if it satisfies the behavioural specification. Given a set of events in the behavioural specification ($\dot{BS}$) and the predicted behaviour (B), the verification task is to detect the difference between the two sequences of events. Notice that in this way we can verify not only 'what' has been achieved, but 'how' it was achieved. If there is no discrepancy then we can conclude that the current design within the provided scenario satisfies the specification. Otherwise, the design needs to be modified.

Modification. Once the discrepancy is detected in the verification phase, the modification phase tunes the variable components among design elements. The candidates to be brought to modification are design D and the modifiable components in the environment E. Since E contains knowledge about physics as well as initial configurations of components, modification to E applies only to controllable components of this element.

The following chapters 6 and 7 describes the framework for design support based on behavioural specification of devices.

5.5 Summary

In summary, this chapter has discussed the characteristics of design task and how causal reasoning can contribute to model the design process. The major points are:

- Design is an ill-structured problem where there is no infallible process which we can follow. However, the preliminary analysis in the early stage of design can

reduce the complexity of the problem to a certain degree.

- The preliminary stage is where there is not yet clear picture of how the design should be. The design problem is still being explored and a rapid generate-and-test cycle is required. Causal reasoning enables the designer to capture the rough picture of the design task.
- An abduction model of design is advocated for the formal description of design in the causal reasoning framework.

This prepares the environment for behaviour-oriented design which focuses on the behaviour of the device rather than its static constraints. We see in the next chapter the notion of behavioural specification and its formulation.

Chapter 6

Behaviour-oriented specification and verification

The function of an artefact describes its role and purpose, and it is considered to represent the behaviour of the artefact intended by the designer. Building an object or device is often motivated by the necessity for a particular behavioural function which is missing in the world and the identification of such a need is an important factor which stimulates design. This standpoint forms a reasonable justification for representing a device by its functions. For instance, a door-bell has a function of notifying a person inside the building when a visitor arrives at the gate and operates a trigger outside. Functioning of a device can be captured by the events or states which are changed (or in some cases unchanged) by its application. In the real world, certain events call or trigger a function which 'causes' subsequent events to happen, and this can be treated, in a broad sense, as a causal relation. For the case of the door-bell, the operation of the trigger (which is often a switch) 'causes' attention of the person in the building to be drawn.

One of the earliest attempts for representation and reasoning about design in terms of functions is by Freeman and Newell[71]. They suggested an approach of representing physical devices as functions and demonstrated its application to the configuration task by focusing on what devices should do. More recently, Barrow[84] introduced a formal method of verifying the correctness of digital circuits by comparing the intended and actual behaviours. Other related research is described briefly in later sections.

This chapter focuses on the behavioural aspect of design and verification. The idea put forward here is the application of logic CI for representation and reasoning about functions in design. Since design tasks here are seen from the behavioural point of view, the types of physical devices which draw interest are dynamic and have significant temporal aspects. Also an emphasis is placed on the feasibility of the design from its functional point of view and not optimisation. Readers may refer to [Nakata 92a] or [Nakata 94a] for a concise description of behaviour oriented specification and design.

6.1 Specification in design

Design specifications are the constraints which ensure that the functions required by the designed object are performed as well as possible. They are generated by the necessity to achieve a certain purpose which should be provided by the designed object within particular limits.

Figure 6.1 describes a well accepted framework for design suggested by Gero[90]. In this framework, analysed behaviour is derived from the structure of the object, which is compared with expected behaviour through evaluation. Structure is synthesised from the function which is given by the designer. This framework clearly places functional requirements above structure.

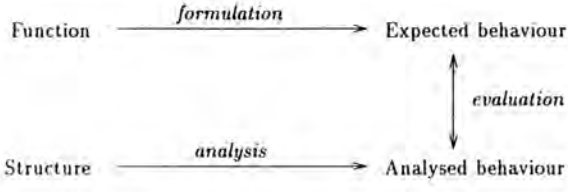


Figure 6.1: A general framework for design(after [Gero 90]).

The relationship between function and structure is in practice more complicated, because once structure has been partially decided, it imposes constraints on the performance of the object. This creates iterative cycles among processes.

Gero's model assumes that the specification of function provides the design goal, which is to construct a structure which engenders the expected behaviour. The generation of expected behaviour is through *formulation*, and this is where designers put much effort. On the other hand, the evaluation is performed between expected and analysed behaviours, which are the projections of function and structure respectively. If the primary purpose of design is to achieve the required behaviour, it is more reasonable to assign the required behaviour as the specification of design. This supports the idea of behavioural specification as discussed in the following sections.

Specifications are usually *minimal*, i.e., there is nothing in the specification that can be removed without altering the performance or the material constraints of the object. This means that what is specified is a necessary condition to be achieved by design. Failure to comply with the specification does not provide solutions to the design problem. On the other hand, the specification itself should be subject to change depending on its achievability. This is one of the characteristics of design problems: specifications are often altered because it is impossible to fulfill them, or because they are ill-structured [Lawson 90].

6.2 Design and functions

The literature most commonly cited in research on function oriented design is that of functional representation by Sembugamoorthy and Chandrasekaran[80]. It introduces *functional representation*, which provides a formal representation for the functioning of a device in many levels of abstraction. Each level recursively describes the functioning of a device in terms of the abstractions of, and relations between, its components. At each level, there are five significant aspects to an agent's functional knowledge (extracted from [Sembugamoorthy & Chandrasekaran 86]):

- STRUCTURE: This specifies relationships between components, and abstractions of these components from lower levels.
- FUNCTION: This specifies WHAT is the response of a device or a component to an external or internal stimulus.

- BEHAVIOR: This specifies HOW, given a stimulus, the response is accomplished.
- GENERIC KNOWLEDGE: Chunks of deeper causal knowledge that have been compiled from various domains to enable the specification of behaviour: for example, a specialised version of Kirchhoff's law from the domain of electrical circuits.
- ASSUMPTIONS: Using which the agent chooses a behavioural alternative over other possible ones.

For example, the functional specification of a device, in this case the function *buzz* is described as below:

FUNCTIONS:

buzz: TOMAKE buzzing (buzzer)

IF pressed (manual switch)

PROVIDED assumption1

BY behaviour1

stop-buzz:....

END FUNCTIONS

Similarly, the structure of a device is described in the way which components and their relations are explicit. Also the abstractions of the components are specified, so that it can make use of other (existing) components.

The *behaviour specification* of a device describes the way in which the overall function of the device is achieved, by composing the functions of the components. This specification can make use of other behaviours, functions, generic knowledge and assumptions. For instance, the behavioural specification of a buzzer can be described as Figure 6.2.

Given a complete set of functions, assumptions, etc., it is possible to model the behaviour of the device by following each STEP. The compiled behaviour would have pointers to the knowledge, functions, assumptions etc. used in the process. Since there could be more than one way to achieve the behaviour or function required, we can suggest alternatives for achieving the behaviour.

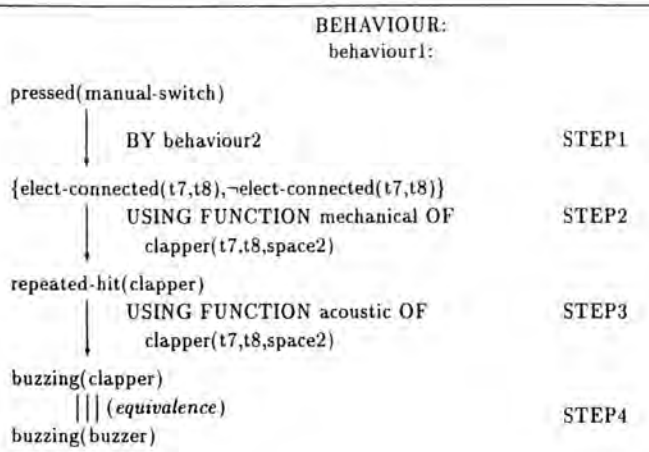


Figure 6.2: The behavioural specification of a buzzer [Sembugamoorthy & Chandrasekaran 86].

Essentially, the functional representation organises the agent's (human) understanding of how the device's *structure* makes it possible to achieve its *function* from the *behaviour*, and exposes the pointers to *generic domain knowledge* and *assumptions about behavioural alternatives* used by the agent in this process. An application to the diagnostic task is suggested in the paper.

Sturges[92] describes a model for conceptual design in which "functional representations"[1] express a design in function terms and are capable of manipulating this representation (function logic). *Function* is defined as "largely domain independent characteristics or behaviours of elements or groups of elements of a design". The *intent* of a design is expressed by the totality of its functional elements and their structures.

The high level description of a function would be transformed into a Function Block Diagram (FBD). An FBD consists of *function blocks* (or nodes), which contain the function name expressed as a generic noun/verb pair to describe the function of the

[1] "Functional representation" here does not refer to that of Sembugamoorthy and Chandrasekaran.

product or process, and *links* which connect higher level functions with lower-level functions. Links suggested are how/why, causal, temporal, informational, alternative, revisional and chaining. An FBD typically looks like a tree, from higher-level functions to components which become leaves. FBDs can be synthesised by function logic, which is based on the construction of appropriate noun/verb pairs. Several FBDs can be merged to represent a new function. There are also conversion rules for FBDs to control block diagrams (CBDs) (and vice-versa) to enhance the expressiveness of the representation, such as to express feedback loops.

Sturges concludes that what this decomposition process in function logic suggests is essentially a reasoning structure which relates each component to the basic function in the design. It is claimed that the enhancement to function logic can be made by adding new vocabulary (links) for design.

While these two approaches take different representations and meanings of functions, the common idea is that function should be the driving force of design process, and the goal of design is to achieve a desired function. The essence of the proposed framework is the same but takes a different view of the design goal and its representation. What is seen as the design goal is to achieve a desired sequence of events of the system as a whole, i.e., including both the artefact and its surroundings (recall the discussion on "inner" and "outer" environments in Chapter 5); and it is best represented by the sequences of events which are desired to happen.

The following subsections examine three issues worth considering in the functional approach to design.

6.2.1 On "function" and "behaviour"

When discussing the behavioural aspect of an object, it is important to notice that various papers use terms "function" and "behaviour" in different senses. Papers based on Sembugamoorthy and Chandrasekaran's functional representation distinguish the two concepts as follows. They treat "function" as some kind of black-box, which only specifies the input and the output, while "behaviour" is the way things happen. As is

formalised in the functional representation, behavioural specification makes use of other behaviour and functions, which suggests that functions are the lower level description of a behaviour. Sembugamoorthy and Chandrasekaran state their reason for their distinction between function and behaviour [Sembugamoorthy & Chandrasekaran 86]:

> In our approach, however, the function is distinguished from the behaviour of the device to accomplish it. Our functional representation is a hierarchical organisation of behavioural segments of components into a representation which itself is not a causal sequence of partial states, but can be processed to obtain such a sequence.... The hierarchical nature of the functional representation is very important; otherwise describing the functioning of a complex system would involve an excruciatingly long sequence of causal states at a low level of abstraction. Simply we distinguish an agent's description of the causal/behavioural sequence device undergoes from the representation used by the agent to produce such a description, and identify the latter with the functional representation or the mental model of the agent.[p.66]

On the other hand, Sturges' definition of function *includes* behaviour and a function can be analysed by its *internal* behaviour, suggesting that behaviour is the lower level description of a function. There seems to be no agreement on the distinction between the two concepts.

The distinction of these two concepts relates to the representation of two important aspects of the performance of a device—'what' it should perform and 'how'. Sturges argue that the higher-order functions describe 'why' part of knowledge and the lower-order (towards the leaf of the FBD) functions describe 'how'. 'Why' part of knowledge in Sturges' sense suggests 'what' is achieved, since a query 'why is this device necessary?' would be 'to achieve something', and this 'something' is 'what' is being achieved. On the contrary, Sembugamoorthy and Chandrasekaran see 'how' a device works as an aggregation of 'what' is being achieved. The difference here is perhaps what is seen as the primitive of the performance of a device. Sturges sees the 'how' part of knowledge

to be the primitive concept, while Sembugamoorthy and Chandrasekaran find it in the 'what' part of knowledge.

The proposed framework treats these concepts in the following way. Behaviour is a sequence of events which is provided by a performance of a device. It is descriptive and can be represented as a chronological description of events. Function is what produces such behaviour by means of relating certain events to invoke other events. It is interpretive and such relation can be represented as causal relations, since it describes what follows what (hence behaviour). If we can construct functions by means of causal theories, behaviours are the sequences of events obtained via model construction from a causal theory. In this framework, 'what' is achieved can be interpreted from looking at events at various time points; we can see 'how' it is achieved by looking at the sequence of events that lead up to it. This suggests that the relation between a function and a behaviour is not hierarchical as previous works suggest. They are looking at sequences of events from different perspectives.

6.2.2 Temporal representation

None of the papers summarised above has separate inference schema or representations to handle temporal information. They all assume that the sequence of events or state transitions subsumes the temporal ordering of events. They all lack the ability to deal with temporal intervals. For instance, to specify that a switch would go on for a certain period and go off later is difficult in KRITIK2 [Stroulia *et al* 92] or Function Logic. DME [Iwasaki & Chandrasekaran 92] provides a facility to set up a "clock", which can begin from 0 and increase (with a positive qualitative derivative) which can cause certain events on its way (e.g. sunrise, sunset, etc.). This facility, however, does not go beyond the labelling of transition states in qualitative simulation.

Functional representation, KRITIK2 and Function Logic all suggest the need of some kind of temporal reasoning. Although these may have implicit notion of time, it is certain that the manipulation of temporal knowledge would enhance their expressiveness. Representation such as Shoham's temporal logic which we explored in the earlier chapters is more expressive in terms of temporal specification with explicit temporal

terms.

6.2.3 Representing defaults

One of the features Shoham's logic and its extension provide which is not possessed by the other systems above is the specification of defaults. Although the design problem typically does not specify every single parameter in detail, all of the systems above require very specific knowledge on their simulation/formulation. This difference, of having defaults or not, results in the difference in the number of possible worlds to consider. Due to the "unique cmi theorem", logic CI will always have only one model, which is "most likely to happen". On the other hand, systems such as DME deal with multiple worlds, mainly because of the ambiguity in its qualitative reasoning mechanism.

The question is, "which is more useful?" How do human designers handle this? It is true that given a very weak constraint, there should be more than one possibility. Sometimes it is more useful to see all possible outcomes, from which the designer can choose. The reason why I think unique model is appropriate is that it gives a flavour of how a device would behave by default. If the designer is satisfied with the outcome, maybe using the default is good enough; if not, s/he can find out which default to override. In case of multiple worlds, we can see what *can* happen, but sometimes the number of outcomes is too high to manage. The process of choosing the "right" one would nevertheless be the same.[2]

6.3 Specification by behaviour

Let us return to the formula 5.2(p. 107). In this abduction model, the design task was to construct design D from other elements. Among these elements, environment E is given by the domain specification since it is the knowledge about the physical properties and heuristics about the world, and the domain specific knowledge. Scenario

[2]In this context, I am assuming that a device would behave uniquely under given conditions. There are cases when a device *should* have multiple behaviour, but I only think about the cases when the design is under-constrained.

S is provided as the boundary conditions for individual cases. Behaviour B is modelled from E, S and D.

What needs to be specified in this framework is a set of events which reflects the necessary behaviour of the device and is compared to the modelled behaviour B. This is the behavioural specification BS, which is defined as follows.

Definition 15 (Behavioural specification) *The* behavioural specification *of a design is, given a scenario, the sequence of events and states which are desirable and necessary to the performance of the artefact. Formally, a behavioural specification BS is a set of atomic formulae φ with temporal ordering.*

Intuitively, the behaviour of a system can be depicted by a sequence of events, which together with the events in the surrounding environment constitutes a *history*. The idea is to specify a desirable history for the system for a scenario by means of events that are necessary to provide such history. It is important to point out that it is not only the history of the device, but of the whole system. We should take into account the changes in the environmental conditions the operation of the device may impose. For instance, a door-bell should not only ring, but should be noticed by those inside the building.[3] The recognition of the bell ringing is beyond the behaviour of the door-bell itself, but it is an important effect of the operation of the device. The events specified in the behavioural specification should be totally ordered over time. We must specify the time point or a time interval where an event is taking place.

Below is an example of a behavioural specification for a door-bell.

$$BS = \{(1,1,switch(on)),(2,3,ring(bell)),(4,4,notify(person(inside)))\}$$

Each event is represented as $(T_{begin}, T_{end}, Event)$, where T_{begin} and T_{end} respectively denote the initial and terminal time points in which the $Event$ takes place. The behavioural specification above reads in English, *"(We want something that when we) turn on the switch it rings the bell for a duration of time and notifies the person inside*

[3] Assuming that there is somebody in the building.

the building". For simplicity, appropriate integers are assigned to the time points which designates the total ordering of the events. However, this does not prevent us from assigning temporal symbols instead of integers, along with their temporal orderings.

6.4 Representation by extended CI

In summary, each of the elements E,S,D,B and the behavioural specification BS are represented in the language CI as follows.

Environment

The laws of physics and causal knowledge in the domain are represented as causal rules. An environment comprises of sets of causal theories each representing a set of knowledge for a particular aspect. For instance, a domain for logic circuits comprises causal theories for AND-gates, OR-gates, NAND-gates, etc. The relation between causal theory for environment Ψ_E and an environment E is $\Psi_E \subset E$.

Scenario

A sequence of events which is assumed to occur is represented by base terms and potential terms. These comprise the boundary conditions for causal theories. This is described in detail in the next section. A scenario S is a set of these terms, i.e., $\Box\varphi, \text{POTEN}\varphi \in S$.

Design

Each component and the causal connections as the results of certain configuration are represented as causal rules. The relation between causal theory for design Ψ_D and a design D is $\Psi_D \subset D$.

Behaviour

Once the causal theories for E,S and D are merged together to form a larger causal theory, it would have a unique cmi model for prediction. This model represents the behaviour of the combined causal theory which represents the current design under a

particular scenario. In this case, the unique cmi model M is equivalent to the behaviour B. i.e., $M \equiv B$.

Behavioural specification

Behavioural specification is a set of desired events in a total temporal order specifying their sequence. Each event is an atomic base formula, such that for a behavioural specification BS, $\varphi \in BS$. Given a behaviour B, the purpose of design is, ideally, to achieve $BS = B$.

To match corresponding elements in Sembugamoorthy and Chandrasekaran's work, we can see the mapping described in Figure 6.3. Structure is what we require to achieve as design, which may or may not include some components which act as functions. All the assumptions can be described in terms of scenarios since they are essentially boundary conditions for the causal theory. One of the main differences between functional representation and the proposed representation is that while the former has hierarchical and functional structure in the representation of behaviours, the latter is purely event based. The event based representation is simple and yet expressive enough to specify the necessary behaviour of a device.

6.5 Scenario

We now take a look at S, the scenario of the system. A scenario is a sequence of events which are assumed to be facts. In causal theory, it is a set of boundary conditions, and the events in a scenario are base formulae and potential terms. It can be defined as follows.

Definition 16 (Scenario) *A scenario is a set of events or setups which represents various possible cases in the domain and acts as the boundary conditions of a specific case. It describes facts and assumptions that hold specifically for such case for a prediction purpose.*

A scenario in the door-bell example would appear as

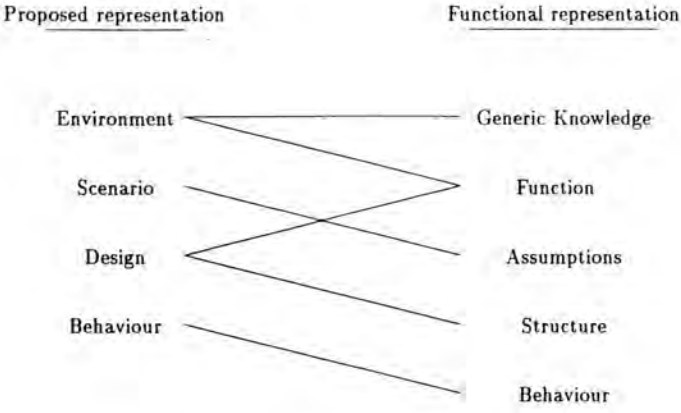


Figure 6.3: Correspondence between *functional representation* and the proposed representation.

$$S=\{\text{POTEN } (1,\infty,\text{p-}powerSupply(on)),\Box(1,\infty,deaf(person(inside)))\}$$

and it reads, *"the power supply is assumed to be on unless it is turned off, and the person inside the building is deaf."*[4]

The reason why we treat scenarios and environment separately is that while environment refers to the knowledge or facts which are always true in the domain, a scenario only describes the events which are assumed to happen to test for a particular case. This is analogous to the relation between the set of physical equations such as those of Maxwell and the values for the parameters in those equations such as electrical currents. This separation contributes to the modularity of knowledge. Assuming that there might be more than one possible behaviour of a device depending on the conditions we set up, we can test them by merely replacing the scenario without altering other elements.

[4]The reader should notice that in this scenario, person who is being notified is deaf and therefore ringing the bell for the purpose of notifying does not work. If we did not consider about the "outer" events, this deficiency cannot be found.

6.6 Design verification

Once the prediction of the behaviour is obtained (we will call this "predicted behaviour"), the task now is to see if it fulfills the behavioural specification. Given a set of events in the behavioural specification and the predicted behaviour which is the set of events in the cmi model, the design verification is to detect the difference between the two histories. If there is no discrepancy between the two, we can conclude that the current design with the provided scenario satisfies the specification. Otherwise, the design needs to be modified.

We first examine some related work on design verification based on behaviours and describe our approach later.

6.6.1 Verification based on behaviours

DME

Iwasaki and Chandrasekaran[92] focus on the task of design verification based on both knowledge about the structure and the intended function of a device. The main issue here is "the question of when one can say [that] a behaviour predicted by a prediction system achieves the desired function in the manner intended by the designer".

Their definitions of "function" and "behaviour" are as follows: *Function* is WHAT the device is supposed to do, and *behaviour* is HOW it is supposed to achieve the function.

They list three criteria for a behaviour of a designed artefact to meet the desired goal:

1. the overall function of the device is achieved,
2. the expected chain of events happen in the predicted behaviour, and
3. the causal connections expected between events exist in the predicted behaviour.

Their prediction of the behaviour is conducted by DME (Device Modelling Environment), developed at Stanford, which is based on QSIM [Kuipers 86] and causal ordering theory by Iwasaki and Simon [Iwasaki & Simon 86a].

The specification of the design is based on the functional representation by Sembugamoorthy and Chandrasekaran [Sembugamoorthy & Chandrasekaran 86]. A function F is defined as a quintuple
$\{Type_F, P_F, Dev_F, C_F, G_F\}$, where

$Type_F$: One of $\{ToMake, ToMaintain, ToPrevent, ToControl\}$.

P_F: The functional goal, i.e., the wff that the function is to make true.

Dev_F: The device that this function is a function of. This has to be a model fragment in the DME's knowledge base.

C_F: The condition which specifies when the function must be achieved.

G_F: The set of CPD's (Causal Process Description) describing the causal mechanism to achieve the function.

In order to meet the desired goal, C_F and G_F must meet the predicted behaviour produced by DME. In particular, an emphasis is placed on the existence of appropriate causal links between the events. The example shown in the paper is the electrical power system (EPS) aboard a satellite orbiting the earth. The research is now at the implementation of automatic verification of the realisation of behaviour specified in the functional specification.

KRITIK2

Another work using functional representation of devices for verification task is the system KRITIK2 by Stroulia *et al.* [Stroulia *et al* 92]. The task here is *blame-assignment*, which is defined as follows: Given a system (e.g. a plan, a program, a problem solver) that fails to deliver the behaviours desired of it, the general blame-assignment task takes as input a specification of the structure of the system, the desired behaviours of the system, and the behaviours delivered by the system. It gives as output a specification of the structural faults responsible for the failure of the system to deliver the desired behaviours.

In this system, *function* refers to the "transformations from the input state to the output state" represented by a schema, while *(causal) behaviour* is the sequence of substance state and component state changes caused by the interaction of the substance flow and the components. *Behavioural states* are also represented as schemata, and they are linked by *state transition schema*. These with *design schema*, *structure schema* and *functional specification schema* constitute the *Structure-Behaviour-Function Model* (SBF Model). Behaviours are typically treated as "internal causal behaviours", which convey the internal causal transition of parameters and features, as opposed to the function which states the overall input and output.

KRITIK2 has an elaborate algorithm for the detection of the differences between the desired and actual behaviours. Since the behaviours are represented as schemata, the differences are detected by comparing the values (or assignments) in each slot of the schemata. An example of a "behavioural state schema" is shown below:

behavioural state:(

previous: previous state
next: next state
enabled-by: preceding state-transition
substance-schema: substance description at current state:

location
(property value) {contained substances description})

Notice that this schema provides a frame-based representation with slots for representing behaviour (previous/next states) as well as for entities (substance schema). It is an expressive representation, and this approach can be useful in comparatively complex domains.

There are two main modes in blame assignment task; design adaptation and redesign. In case of design adaptation, the inputs are the specifications of the function desired and that delivered by the design, and the output is a specification of the structural element(s) in the current design which is(are) responsible for the failure of delivering

the desired function. For redesign, the inputs are the specification of the undesirable behaviour that the design exhibits and the structural element in which the undesirable behaviour is observed, and the output is a specification of that structural element which is responsible for the side-effect (undesirable behaviour).

While DME and KRITIK2 both suggest the comparison of schema-like structures, the verification process proposed in our framework is event based. The behaviours represented in this way are easily accessible and intuitive. The following subsections describe verification by comparing event based representation of behaviours.

6.6.2 Verification by discrepancy identification

In the proposed framework, the verification is performed in the following way. Once the prediction of the behaviour is obtained (we will call this the 'predicted behaviour'), the task now is to see if it fulfills the behavioural specification. Given a set of events in the behavioural specification and the predicted behaviour which is the set of events in the cmi model, the design verification is to detect the difference between the two histories. If there is no discrepancy between the two, we can conclude that the current design with the provided scenario satisfies the specification. If discrepancies exist, then the design should be modified.

6.6.3 Detection of Discrepancies

Discrepancies between two histories arise when the two have contradictory or different sequences of events. For instance, if the sequence of the events (propositions) p, q, r is actually $p—q—r$ in one history and $p—r—q$ in the other, the two histories can be considered to have a discrepancy. There is always a history that acts as a model (*behavioural specification*) and one that is to be verified (*predicted behaviour*).

We begin with identifying the basic discrepancies. As we discuss later most discrepancies are expected to be a combination of the basic cases. The description of the basic cases in the propositional case is given first, and then the first order case is discussed.

6.6.4 Basic Discrepancies

There are typically three basic discrepancies here: 1) insufficiency (propositions missing), 2) redundancy (have extra undesirable propositions), and 3) divergence (different histories). These cases are described in Table 6.1. In the first two cases, the histories compared are basically the same but with some parts being inconsistent. Precisely speaking, there are two distinct cases in the third case: propositions q_n being irrelevant to p_n (which is equivalent to having '—' in their places) and q_n being $\neg p_n$. It should be noted that the time points (t_n) are not absolute, but relative: they merely describe the order in which these events occur.

These histories are illustrated in Figure 6.4. Notice the difference between the redundancy case and the divergence case: the former refers to the case where an isolated event introduces inconsistency between the history and the model; the latter is the case when the history does not converge with the model within the time scope of comparison.

Table 6.1: Three basic cases of discrepancies: *insufficient* — a proposition missing, *redundant* — an extra proposition, *divergence* — histories diverging.

MODEL	insufficient	redundant	divergence
(t_1,p_1)	(t_1,p_1)	(t_1,p_1)	(t_1,p_1)
(t_2,p_2)	(t_2,p_3)	(t_2,p_2)	(t_2,p_2)
(t_3,p_3)	(t_3,p_4)	(t_3,q)	(t_3,q_3)
(t_4,p_4)	(t_4,p_5)	(t_4,p_3)	(t_4,q_4)
(t_5,p_5)	—	(t_5,p_4)	(t_5,q_5)
—	—	(t_6,p_5)	—

In the first-order case, the basic discrepancies are the same as the propositional case except that the comparison of the events involves a unification process. For instance, if the next event occurring in the model is $value(1)$ and the prediction gives $value(X)$ (where X is a variable), these two events are consistent provided that the variable X has not been instantiated to another value in the previous time point. Also owing to the semantics of the first-order logic, two terms are inconsistent not only when they are contradictory by negation (e.g. $p(a)$ and $\neg p(a)$) but semantically. For exam-

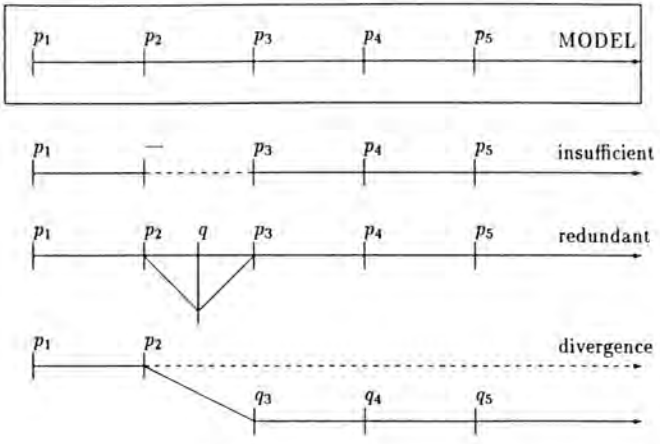


Figure 6.4: Three cases of basic discrepancies.

ple, *value(device1,output(+))* and *value(device1,output(−))* representing the outputs of *device1* which are contradictory.

6.6.5 Analysis of discrepancies

Discrepancies between the behavioural specification and the predicted behaviour need to be analysed to identify the time points which require modification. It makes the task easier if the discrepancies are actually sorted into the basic cases. Since we deal with discrepancies point-wise, i.e., starting from the earliest time point until we encounter the first discrepancy, we can concentrate on the short sequences of events rather than the complete history.

Comparison of the histories

As mentioned earlier, since the discrete time points attached to the events have no significance other than the definition of the ordering among the events, the sequence of events should be compared by their relative orderings. The basic strategy is to start

from the earliest time points of two histories, and proceed chronologically until the first discrepancy is found. Since the behavioural specification consists of the *necessary* sequence of events, all events in it should appear in the predicted behaviour. On the other hand, we can skip some of the intermediate events in the predicted history which do not appear in the behavioural specification as irrelevant or unimportant events. If the predicted behaviour contains all the events comprising the behavioural specification, we can judge that we have achieved the task. Otherwise, when the predicted behaviour misses out an event, we can detect the discrepancy between the two.[5]

Combination of basic discrepancies

If there is more than one rule missing, or extra, more than one basic discrepancy is anticipated. Generally these discrepancies can be considered to be combinations of basic discrepancies. For example, the predicted sequence of events might be

$$p_1 — q — p_3$$

when the expected sequence is

$$p_1 — p_2 — p_3.$$

The predicted one can be seen as 1) missing the event p_2 and 2) having an extra event q. In this way, we attempt to break down various cases of discrepancies into basic cases. In the design modification stage described in the following chapter, each basic discrepancy can be dealt with separately, simplifying the task.

The basic algorithm to detect the discrepancies is fairly simple. Essentially, we take the events in the behavioural specification in chronological order to match them with the events in the predicted behaviour. Since behavioural specification is minimal, we only need to see if every event in the specification is achieved in the prediction. Figure 6.5 describes the algorithm to identify the earliest discrepancy.

When two sets of events $E_{t_{BS}}$ and $E_{t'_{PB}}$ are *compatible*, it means that

[5] In cases where such intermediate events are not specified explicitly, we can assume that the specification 'doesn't care' how the succeeding events should be achieved.

```
BS ← Behavioural specification;
PB ← Predicted behaviour;
t_BS := 0; {time step for BS}
t_PB := 0; {time step for PB}
E_tBS ∈ BS; {events at t_BS in BS}
E_tPB ∈ PB; {events at t_PB in PB}

begin
    while t_BS is smaller or equal to the ltp of BS
        Compare E_tBS and E_tPB;
        if they are compatible then
          increment t_BS and t_PB by 1;
        else
          Retrieve the events E_t+1BS;
          while   t_BS is smaller or equal to the ltp of BS
                increment t_PB by 1;
                if E_t+1BS is compatible with E_tPB then
                  E_tBS is insufficient;
                  return;
                else if E_tBS is compatible with E_t+1PB then
                  E_tPB is redundant;
                  exit this while loop;
          end
          The histories are divergent at t_PB;
          return;
    end
    No discrepancy;
end
```

Figure 6.5: The algorithm to identify the earliest discrepancy.

- every event in $E_{t_{BS}}$ is present in $E_{t'_{PB}}$, and
- there are no inconsistent events among the two sets.

By this algorithm, if there is a behavioural specification of p_1—p_2—p_3 and the predicted behaviour of p_1—q—p_3, it detects that p_2 is *insufficient*; once the causal theory is modified and achieves p_2 in the predicted behaviour, then it detects that q is redundant. This means that the priority is given to the insufficiency over redundancy. This is justified by the fact that the specifications are minimal and all the events in the behavioural specification must be fulfilled.

One of the drawbacks of this method of breakdown into basic discrepancies is that it could miss out some important and potentially useful discrepancies. For instance, events simply being out of order, e.g., p—q—r and p—r—q, is quite common, and provides a basis for more elaborate examination. However, as it stands, this will be labelled as insufficient or redundant. It is desired that such cases are added to the set of basic discrepancies.

6.7 Summary

This chapter described the specification of design in terms of the behaviour of the designed object, and a method to verify the object's performance according to the specified behaviour. In summary the following points are raised:

- Specifications in design are essentially functional and the purpose of design is to achieve a particular behaviour of an object with appropriate functions. Behavioural specifications identify the desired behaviours to be achieved by the designed object.
- In the proposed abduction model of design, *scenarios* provide sequences of events which are assumed to happen in various cases given same environment and design.
- Causal knowledge is represented by causal theories in extended CI which provides an executable representation and a unique model which is a plausible prediction

of the behaviour of the system.

- Design verification is conducted by comparing behaviours which forms the basis of the behaviour-oriented approach to design.

Given the representation of the design and modelling of its behaviour, the next chapter proceeds to the modification phase of design.

Chapter 7

Design synthesis

As we have seen in the previous chapters, the purpose of design is to construct a structure which achieves desired behaviours. This is constructed through a cycle of modification and evaluation.

In the proposed approach, structures are represented as causal theories. Causal theories provide the bases of predicting the behaviours of the system, and the emi model of a causal theory provides its most plausible behaviour. There are a number of choice points in the modification phase: many syntactically legal modifications may be possible to fix the discrepancies between the desired and predicted behaviours. Choices of plausible modifications need to be guided by criteria, which can be regarded as optimisation measures. Optimisation measures are often implemented by setting certain criteria to guide design. In general practice, such preferences as well as constraints decide which design to choose among a number of candidates.

This chapter discusses how design synthesis is carried out by abduction of causal rules, and the consistency maintenance of that process. It also describes the guidance of modification by applying a similar technique we suggested for dealing with reconstruction tasks in Chapter 4 and how it contributes to the optimisation of design.

7.1 Design modification

The result of design verification is forwarded to the modification phase in an attempt to fix any problems that may have arisen. In the "generate and test" style of design, the modification phase contributes to the generation of partial design. What has been modified is tested via the feedback to the design verification phase. Modifications to design become necessary not only during the process of constructing the design from a prototype, but in the re use of past designs to be adapted to the new specification. The main difference between these two activities is that while designing from scratch devotes a larger portion of its process in the development of new components possibly from first principles to construct a prototype, the re use of design involves selecting the most appropriate one from the existing design, or putting together various existing designs and "fine tuning" it. In practice, it is more likely to start a design process by selecting a prototype which is considered to carry features close to those in the specification [Gero 90].

The information necessary for design modification includes

- *which* parameters violates the specification
- *how* the parameter violates the specification, and
- *what* are the constraints associated with the parameter.

Also important factors in this phase are the preference criteria and the optimisation measure (see Section 7.8) to guide the choice of possible modifications.

7.2 Synthesis of causal theories

Since the design is represented as causal rules in the proposed framework, modification of the design is performed by modifying the causal theory. The modification of the causal theory is problematic, since addition or partial alteration of causal rules may affect the soundness and consistency conditions. The maintenance of these conditions is itself a difficult task, but unless these conditions are kept there is no point in employing

representation by causal theories. The reward for maintaining these conditions is prediction by the emi model and the capability to systematically deal with further violations of these conditions.

As we see later, each component of the design is described as a set of causal rules which represents its functions. These sets of causal rules are actually small causal theories which are sound and consistent on their own. The combination or synthesis of components is then performed by merging causal theories to form a larger causal theory, which is also sound and consistent. There are several issues to be considered here.

Transparency. In the proposed framework, a device is represented by a causal theory featuring the causal rules describing its functions. The operation of the device is seen as the events engendered by the application of causal rules. Since causal rules describe the causal relations of events, depending on the level of abstraction it can either be treated as an entity which performs like a "black box", or one whose mechanism (in this case causal rules) is transparent from outside. This decision depends on the level of description necessary in the design process. If there is a need to modify the device itself, it should be transparent; otherwise, if the problem is the configuration of well defined components, each of them may well be treated as a black box.

Vocabulary. As the library of devices grows, the development of causal theories representing their functions become harder since it will be more difficult to grasp the whole range of causal theories involved. This can easily result in inconsistent use of terms, such as assigning different names for the same entity or concept, or assigning the same name for different entities or concepts. Such a problem is more likely to occur if there is more than one person working on constructing the library of causal theories. This problem is less serious if causal theories are completely modular and non-transparent, since it is only the interface to the devices (analogous to input/output predicates) which is necessary to be maintained of its compatibility. However, if causal theories are transparent, there needs to be a fixed set of vocabulary from which the

predicate names and term symbols are chosen in order to guarantee the consistency of the combined theory. Disagreement of vocabulary makes the task of maintaining the soundness and consistency conditions impossible.

Granularity. This issue is related to the choice of vocabulary and addresses the problem of choosing the appropriate level of abstraction. When interfacing two components, the difference in the level of abstraction will cause problems. Unlike the issue in vocabulary, this is not simply an interface problem; it is one of the fundamental issues in knowledge representation. An attempt may be made to develop a tool to support the maintenance of the levels of granularity but it would be on *ad hoc* basis and provides only a partial solution to this issue.

In the actual implementation, causal theories are treated as transparent, with fixed vocabularies and no special treatment is provided to maintain granularity. Issues in granularity and transparency are related to the compilation of devices as discussed later in Section 7.9, but this setup is likely to pose no serious problem in relatively small well-defined domains.

7.3 Abduction of causal theory

Causal theories do not always have a complete set of causal rules. When encountering a situation in which the sequence of events which led up to the present events can not be reconstructed by *abduction over causal rules*, it is often useful to make some amendments to the causal theory. This can be achieved by adding/removing causal rules, or by modifying them. This process can be referred to as *abduction of a causal theory*. It is related to constructing a theory from data by means of abduction. Work in theory construction includes that by Richards *et al*[92] which automatically builds qualitative models from the qualitative behaviours. Their work was motivated by the fact that the development of qualitative models is itself a difficult task.

Suppose, in the glass window problem in Example 3, we know that at time point 7 the glass was broken (*broken-glass*) and at time point 10 there was no glass (*no-*

glass). There is no rule in the current causal theory to conclude this event, so we can speculate that there is a missing rule. Rule 4 says that the glass remains broken unless something happens to make it false, the event *broken-glass* persists until time 9 since further persistence causes contradiction.[1]

Theoretically, if there is an event q at time t_q, and P_{t_i} is a set of events at time t_i such that $t_i < t_q$, any element $p \in P_{t_i}$ can be a candidate for the cause of q. What we assume here, however, is that it is more likely that something which happened immediately before was to be the cause for the event. That is, we should focus on the events in P_{t_q-1} as the cause for q at time t_q.

This suggests that we create a rule

7. $\Box(t,t,broken\text{-}glass) \supset \Box(t+1,t+1,no\text{-}glass)$.

However, this rule is inconsistent with rule 5, that given {*broken-glass*, *replace-glass*}, it concludes the contradiction *glass-window* and *no-glass*. In order to resolve this rule 7 should have a default condition that the glass is not replaced,

7'. $\Box(t,t,broken\text{-}glass) \wedge \Diamond(t,t,\neg replace\text{-}glass) \supset \Box(t+1,t+1,no\text{-}glass)$.

This ensures that the consistency and the soundness conditions be always maintained in order to produce a unique model for a causal theory.

In general, where causal links are missing, when sweeping forward from the earlier boundary conditions or backwards from the latest boundary conditions there should be one or more time points where there exist some inconsistencies. Those are the time points where there are missing causal relations. It is reasonable to link the causal relation at the closest gap, since the causal relations tend to be more intuitively plausible here than taking two events which are chronologically far apart.

It is important to note, however, that this heuristic assumption is rather a strong commitment. By making this assumption, we are rejecting the case where there exists a

[1] We assume *no-glass* as $\neg$*broken-glass*.

stronger causal relation than merely a temporal distance. Especially, spatial distance between objects can be a basis for expecting a causal relation other than the temporal closeness. It is certainly desirable that such knowledge be taken into account in establishing causal links. Another closely related implication of this method is that it makes it impossible to interleave causes and effects involved. Given events a,b,c,d in this chronological order, we cannot establish causal links from a to c and b to d, even if these relations are more causally plausible from a non temporal point of view; this is considerably restrictive. These issues need to be dealt with in the future extensions of this work.

7.4 Constructing design by abduction of causal theories

The abduction of a causal theory essentially constructs a set of rules from causal relations between events which are necessary for a desired history to happen. This process is analogous to the design of a device. Formally, design can be seen as an abductive process (see Section 5.3) and can be expressed by Formula 5.2:

$$E, S, D \models B$$

where E is the environment in which the device operates, S is the scenario under which the device is placed, D is the design of the device, and B is the behaviour of the whole system. The object of design is to compose D from the given E, S and the behavioural specification BS.

From the purely functional view of device and its design, we can represent the knowledge in terms of causal rules. E consists of physical laws and general causal knowledge in the domain, S the boundary conditions, and B a set of events representing the desired behaviour of the system as a whole in chronological sequence. The design, D, is also represented in terms of causal rules which relate to the physical features and configurations of the device, but in their functional form. The causal theory formed by joining E,S and D will have a unique model which represents its behaviour. The design process is an iterative process where D keeps being modified until the predicted behaviour fulfills the specified behaviour. This iterative process constructs a set of

causal rules which represents the functional feature of the device.

This is abduction of D from from B with conditions E and S. Once D is constructed successfully, Formula 5.2 should hold. Abduction is known to be incomplete, and we therefore need to verify the result. This is analogous to the 'generate-and-test' model of design, where generation is abduction and testing by design verification.

7.5 Guidance by causal abduction

The main purpose of abduction framework devised in in Chapter 4 was to cater for reconstruction and interpolation problems using the language of extended logic CI. What can be found by the application of that abduction framework, and by comparison of forward and backward sweeping, are the events which were missing in order to obtain the desired history. If we recall the Stolen Car Problem, it was expected to identify that the 'stolen(car)' event had happened at anytime between the time the car was parked and when it was found to be stolen. Given an event P, which is known to have happened, bidirectional sweeping identifies the range of possible events that should have preceded to produce the causal occurrence of P, based on the assumption that states that potentially persist would do so. Although we do not use the procedure devised in Chapter 4, a similar style of abduction is applied for the behaviour-oriented approach.

This style of reasoning is common to the modification of design based on behavioural verification. In the context of behaviour based design, the central issue in modification is to conform the predicted behaviour to the desired behaviour. When one of the three basic discrepancies, *insufficient*, is detected in the verification phase, the requirement is to make the unachieved event to occur in some legitimate way. This is done by creating and asserting causal rules which enable the causal connection between unachieved events (consequents) and achieved events which precede them (premises). This can be done purely syntactically, merely by looking at the temporal sequence of events. However, the rules created by this procedure are not only very abstract but in most cases implausible. In order to make such rules more plausible, the *causal distance*

between the cause and the effect should be as small as possible. Essentially, an effort is made to close the gap between the cause and the effect by applying the existing causal rules in the domain. This procedure is described in the next section.

Assuming that it is the steps in causal reasoning that decides the distance between cause and effect, we can define the notion of *causal distance* as follows.

Definition 17 (Causal distance) *The* causal distance *between a cause P and an effect Q is the minimal number of causal inference rules that is required to reach the event Q from the event P.*

For instance, if there is a causal rule which describes the causal implication $P \supset Q$, the causal distance between P and Q is 1. If the rules $P \supset P_1$, $P \supset P_2$, and $P \supset Q$ were necessary, then the causal distance between P and Q is 3, since it required three causal rules.

As in interpolation problems, bidirectional sweeping over causal theories contributes to the choice of a plausible modification by identifying the "most likely events" that need to happen in order to obtain desired events. This is merely a heuristic preference criterion but bidirectional sweeping helps to find out what events are actually missing for a successful chain of events between two time points. The following simple example illustrates this process.

Example 6 (Two-way switch problem) *Some of the rooms in an office have two entrances each of which is fitted with a switch to turn on/off the lights. The following scenario can be considered in such rooms. Let* DoorA *and* DoorB *be the two doors to a room. A person entering the room through* DoorA *can turn on the light by flipping* switchA *attached at* DoorA. *When the person leaves the room through* DoorB, *he or she can flip* switchB *at* DoorB *to turn the light off. In short, any flipping event at either switch changes the state of the light (*on *to* off, *or* off *to* on*). The problem is to provide the appropriate configuration of the light and two switches.*

To simplify the problem, suppose the environment knowledge consists of only two causal rules:

E1. $\Box(t,t,circuit(close)) \supset \Box(t+1,t+1,light(on))$.

E2. $\Box(t,t,circuit(open)) \supset \Box(t+1,t+1,light(off))$.

These are augmented by inertial rules that the states *light(on)* and *light(off)* remain in each state unless it is known that certain events (i.e., opening and closing of the circuit) cause the state change. To simplify further, let's assume that we have already connected the terminal a of each switch to the stage illustrated in Figure 7.1.

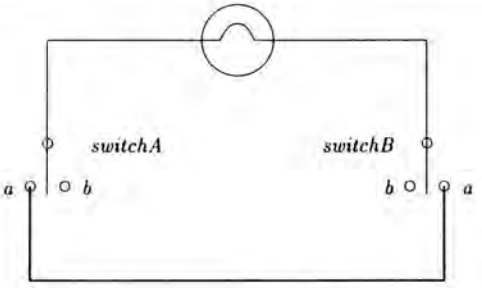


Figure 7.1: An intermediate stage for the two-way switch problem.

The causal rules that describe the relation between switching events and the state of the light in the problem description can be formulated as follows (it already assumes the causal relation between the state of the circuit and the state of the light given in the environment knowledge):

D1. $\Box(t,t,switchA(a)) \wedge \Box(t,t,switchB(a)) \supset \Box(t+1,t+1,circuit(close))$.

D2. $\Box(t,t,switchA(a)) \wedge \Box(t,t,switchB(b)) \supset \Box(t+1,t+1,circuit(open))$.

It assumes two possible positions, a and b, for each switch. Rule *D1* can be derived from the case when the light is turned on, assuming that both switches were in position a. Rule *D2* is derived from the case where *switchB* was operated (i.e., changed its position from a to b), causing the light to turn off.

Now let us consider the following two cases.

- Suppose someone operates *switchB* again and changes its position back to *a*. The light will go on because of Rule *D1*, providing a desirable outcome.
- Suppose someone operates *switchA* and change its position from *a* to *b*. In this case, the light would not turn on because there is no causal rule provided for this to happen. This invokes an interpolation problem that $\{switchA(b), switchB(b)\}$ should result in *light(on)*.

According to our abduction framework, it prefers the event *light(on)* to persist back as far as possible. In order for this event to occur, Rule *E1* must be evoked just before $\{switchA(b), switchB(b)\}$ state (let's call this *S1*) occurred, so there should be a $\Box(t,circuit(close))$ at some point just after *S1*. At this point there are two possible histories. One is using Rule *D1* to make $\{\Box(t,switchA(a)) \wedge \Box(t,switchB(a))\}$ state (let's call this *S2*) happen between *S1* and $\Box(t,circuit(close))$. The other is to assume a causal relation between *S1* and $\Box(t,circuit(close))$.

Since design knowledge is subject to change, the only rule we *must* comply with is *E1* (not necessarily with *D1*). To create the history with minimum number of events, the latter history is preferred—otherwise, more states are necessary to create causal connections between *S1* and *S2*. The solution obtained, *i.e.* the causal rule

D3. $\Box(t,t,switchA(b)) \wedge \Box(t,t,switchB(b)) \supset \Box(t+1,t+1,circuit(close))$,

connects the terminal *b* of each switch to achieve the required behaviour with minimal connections (Figure 7.2).

7.6 Asserting and retracting rules

As we have seen in the previous section, one way of modifying a design is by asserting additional rules to the existing causal theory. Removing some rules where applicable is another way of modifying the behaviour of the device. This section describes how these procedures are performed.

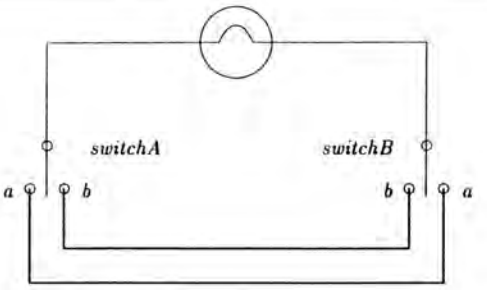


Figure 7.2: The complete connections in the two-way switch problem.

7.6.1 Rule assertion and proof procedure

We begin by asserting a causal rule which would give the desirable history for the current design. This rule is a very abstract rule which may convey no intuitive causal relation whatsoever between the events, but has an effect of 'forcing' an event which is desirable to occur at a time point later in the occurrence of the premise given. Since the requirement is captured as the cause and effect relation, the task is to prove the following style of causal rule.

Prove:

$$\Box\varphi_P \wedge \Box\varphi_T \supset \Box\varphi_Q \tag{7.1}$$

where φ_P and φ_Q are of the form $\text{TRUE}(t, t', p)$ (base formula, p is a proposition or a first-order formula) and φ_T is a formula that describes the ordering of the temporal variables in the rule.

The object of the operation is to rewrite the causal rule (7.1) and prove the rewritten version by existing causal rules. In order to achieve this, several operations can be applied to the rule.

Operations:

- *Antecedent*: Find alternative rule
- *Consequent*: Perform abduction

Essentially, for $\Box\varphi_P$ in the antecedent, find a *causal* rule that contains this formula in its antecedent. If such a rule exists, say $\Phi_{P1} \wedge \Theta_{P1} \supset \Box\varphi_{P1}$, then $\Box\varphi_P \in \Phi_{P1}$. For the consequent, find a *causal* or *interval* rule that has $\Box\varphi_Q$ as its consequent. Assume that such a rule exists, say $\Phi_{Q1} \wedge \Theta_{Q1} \supset \Box\varphi_Q$, then assert a rule of the form

$$\Box\varphi_{P1} \wedge \Box\varphi_{T1} \supset \Box\varphi_{Q1} \tag{7.2}$$

where $\Box\varphi_{Q1} \in \Phi_{Q1}$. Note that $\Box\varphi_{Q1}$ is one of the possible choices from Φ_{Q1}. Be sure that the temporal constraints $\Box\varphi_T$ and $\Box\varphi_{T1}$ are consistent. These operations are illustrated in Figure 7.3.

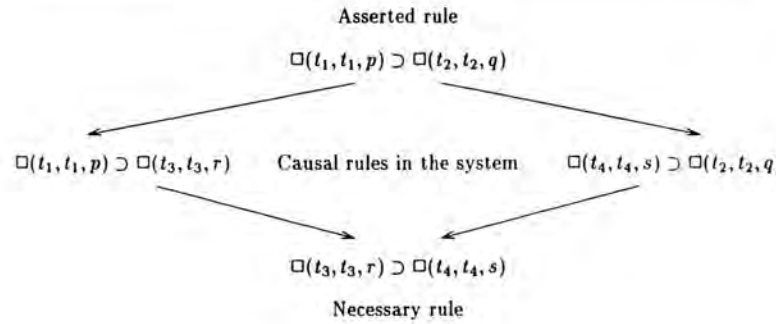


Figure 7.3: A simple proof operation of an asserted rule.

These operations are only applied where applicable. If nothing can be done about, for instance, the antecedent, just leave it as it is. If neither of these operations can be applied, then it is the 'dead end' case (see below).

At this point there are several possible cases:

1. Rule (7.2) resembles an existing rule, as described below.
2. Above operations can be applied further to the antecedent and/or the consequent.
3. Dead end.

The causal rule constructed in this manner does not result in the reconstruction of an existing rule, since if there were such a rule in the causal theory, it would have been applied in the modelling stage. In the first case, the constructed rule would look similar to an existing rule but would not subsume it. For instance, a necessary condition could be missing in the new rule, but with all other conditions in the antecedent and the consequent being the same. In this case, rather than just asserting such a rule, it would be more reasonable to make the existing rule be applicable by arranging for the missing condition to be fulfilled. Another possible case of resemblance is the difference in the default conditions, but this could end up violating the soundness condition. The second case in the list suggests that the system may be decomposed further. Otherwise, the third case, nothing can be done to it, which suggest that there is no such device present in the current design, or that "a device which has such function is necessary". By asserting this rule, we modify the design D and can achieve the required behaviour.

However, in some cases physically implausible or impossible rules may be suggested. In such cases, the user should intervene and reject the suggestion and seek for an alternative. At each assertion or retraction of a causal rule, the user is consulted to check for the appropriateness of the operation. Once the rule is asserted or retracted, the rule base, i.e, the causal theory, is checked for its consistency and soundness before the subsequent simulation. The issues regarding the maintenance of soundness and consistency conditions are discussed in the next section.

This procedure so far gives a partial solution to the problem since the following must be proved at this stage.

- Prove the validity of $\Box\varphi_i \in \Phi_{P1} \quad (i \neq P1)$
- Prove the validity of $\Box\varphi_j \in \Phi_{Q1} \quad (j \neq Q1)$

The proof is by repetitive application of operations for each term, or by consulting its occurrence (i.e. that such event occurs) in the sequence of events within the time-bounded model.

7.6.2 Preservation of temporal constraints

The constraints for the temporal terms are represented in one of the cases below:

1. Temporal constants ($\in I$): numerical ordering applies
2. Temporal variables:
   - Differential description among temporal terms as in $\Box(t+m, t+m, p)$ and $\Box(t+n, t+n, q)$: $t+m \leq t+n$ if $m \leq n \quad m, n \in I$
   - Constraining terms of the form $\Box(t_1 \leq ... \leq t_n)$ (or $\Diamond(t_1 \leq ... \leq t_n)$) where t_i are temporal terms appearing in the rule
   - Combination of the two

Temporal constants are typically assigned initially to the boundary conditions. Differential description is typically used for the potential histories (to set up simulation environments). Since the explicit description of the constraining terms are preferred for causal rules, the preservation of temporal constraints in these terms is crucial.

In any operation over an antecedent or the consequent (i.e. expansion using existing rules) the temporal constraints apply, which preserve the temporal ordering between the temporal variables. It is necessary, however, to maintain the temporal ordering in the rule synthesis.

Let $\Box\varphi_A$ and $\Box\varphi_B$ be the antecedent and the consequent of the new (synthesised) rule, which looks like

$$\Box\varphi_A \supset \Box\varphi_B \tag{7.3}$$

and $t_A = ltp(\varphi_A)$ and $t_B = ltp(\varphi_B)$. The (syntactic) constraint then will be $t_A < t_B$, which should be added as the "necessary" condition. There is an implicit assumption

that there is a well defined temporal ordering between the temporal variables within a base formula; otherwise it is impossible to derive the *ltps* of causal rules.

$$\Box\varphi_A \wedge \Box(t_A < t_B) \supset \Box\varphi_B \tag{7.4}$$

7.6.3 Removing Components

Since each rule in the causal theory for design D represents a function of a component, asserting a rule to the set was analogous to adding a new component to the configuration of design. Similarly, the removal or the replacement of a component are performed by removing one or more rules from the theory.

Retracting a causal rule is equivalent to removing a function from which an undesirable event was derived, and, as a result, detaching a component which was responsible for the behaviour. There are essentially two cases for this.

- The causal rule which concluded the event is retractable (i.e. design component rules).
- The causal rule which concluded the event is *not* retractable (i.e. environment causal rules such as laws of physics).

In the first case, removing the rule solves the problem. It is equivalent to removing a component in the design. An alternative is to keep the rule but suppress it from firing by removing the events which satisfy the premise. Between these two options, the option of removing the rules is more practical. While it may seem drastic since it is analogous to removing a function of a device altogether, the same function can be re-instantiated in a different way (i.e. by other forms of causal rules) if it becomes necessary. However, the implication of removing the rule can be more problematic. In some cases, the rule may have concluded a desirable event elsewhere, which no longer could be supported. It is, however, possible to reconstruct a causal rule which concludes the desirable event but with different premises; this will invoke yet another design problem.

In the second case, since it is not reasonable to remove the rule (unless we can alter the environment), the strategy will be the second of those mentioned in the previous paragraph, to suppress the rules from firing. We first take a look at the causal rule and see what were the events that caused the rule to fire. The aim now is to choose an event among them and prevent it from taking place. By checking which causal rule caused the event, we would have the same task we originally had, but with different events and rules, and at preceding time point. This process continues until we find a retractable rule, or possibly a boundary condition. The soundness and the consistency of the causal theory must be maintained by monitoring the changes in the rules.

7.6.4 Summary of design procedure

The overall process of the process can be summarised by the following algorithm. Assuming that there are n possible scenarios to consider. For scenario S_i ($i \leq n$)

1. Provide behavioural specification BS_i.
2. Construct the prediction model (cmi model) M_i incorporating S_i in the causal theory.
3. Compare M_i and BS_i; detect the earliest discrepancy.
4. If there is *no discrepancy* proceed to the next scenario S_{i+1} and continue from 1.
   Else if *insufficient*, add causal rules to provide necessary event to occur. Then repeat from 2.
   Else *redundant*, add/remove causal rules to suppress the occurrence of the redundant. Then repeat from 2.

Here we assumed only two types of discrepancies for simplicity. Once any modification is made then the new design must be verified against behavioural specifications which have been previously satisfied, since the modification may result in different behaviours. The process terminates when no discrepancy is detected for all scenarios considered. However, termination is not guaranteed: the specification may be too constrained making it impossible to satisfy all behavioural specifications, in which case

the constraints must be relaxed.

7.6.5 A comparison with planning systems

The view of design synthesis presented above shares similar characteristics with some formulations of *planning* problems. In planning, a description of a starting situation is given and a desired goal is specified; the task is to provide a series of actions that achieves the goal from the starting situation. One of the most successful planners was STRIPS [Fikes *et al* 71, Fikes *et al* 72]. STRIPS provided a set of *macro operators* which are generalised pre and post conditions of successful plans with intermediate actions. Essentially, a planning task was achieved by finding a series of operators which transformed an initial state into a goal state. This approach was later made more efficient and general by *nonlinear* planners such as NOAH [Sacerdoti 75], NONLIN [Tate 76] and their mathematical reconstruction TWEAK [Chapman 87]. The advantage of nonlinear planners is that they use *partial plans* which means they can search through the space of partially ordered actions instead of complete plans which reply on total ordering of actions; by using partial plans, certain decisions can be left until execution time thus reducing the search space significantly.

Briefly, here is how TWEAK works. The user supplies TWEAK with a set of partially instantiated plan templates called *steps*. Each step corresponds to a causal axiom with pre- and post-conditions. The plan is constructed by a partial ordering of steps and their variable bindings. The initial problem description is an incomplete plan in which there is no step between the initial situation and the goal situation. A step whose post conditions match the initial situation and another step whose pre conditions match the goal situation are selected constructing the first incomplete plan. TWEAK then iteratively searches through the set of steps to complete the plan by selecting appropriate steps that establish the causal link between the initial and the goal situations. During this process, an addition of a step can be seen as a transformation from a partial plan to another. There is a step in TWEAK which avoids 'clobbering' (an insertion of an event preventing the pre conditions of the following step) thus

enabling it to deal with the Sussman Anomaly.[2]

To compare with the design synthesis described here, the starting situation and the goal in planning can be seen as the conditions in the initial and terminal time points in a given scenario. The task in design synthesis is to find a set of causal rules which provides a causal sequence of events as specified in the specification. The nature of the problem is very similar to that in planning.

Some of the differences between the two approaches can be summarised as follows. In planning, all the valid actions are strictly predefined, and the task is to identify which actions to take in which order. This is analogous to the configuration problem, where the task is to find what configuration of components achieves the desired functionality. In design synthesis, some of the causal rules are literally synthesised in order to achieve a missing causal link, and a new component whose requirement is to provide such a causal link is identified.

Another difference is that the process of design synthesis is model based, i.e., a necessary or redundant component is identified by comparing the desired behaviour (behavioural specification) and the predicted behaviour. However, even in nonlinear planning the method taken is based on iteratively searching for actions which achieve the preconditions for the goal leading to the initial situation. Although this method has proved to be effective in planning, the model based approach suits better in design where some desired functions need to be identified.

Despite these differences, there are many common features in the nature of the two tasks. The issue regarding the removal of rules which may have concluded desirable events elsewhere (as discussed in Section 7.6.3) can be seen as 'protection violation' in planning, and a technique to get around this problem can be found in planning research (for instance [McDermott 78]). An investigation into the use of techniques in planning research in this context of design is essential in the future.

[2]This is a situation in the blocks world where linear planning fails. Given the description of a stack of blocks as the conjunction of *on(a,b)* and *on(b,c)*, no matter in which order we try to solve these subgoals, the first subgoal must be violated in order to solve the second.

7.7 Conflicting defaults

Asserting new causal rules may bring in some undesired side effects—violation of consistency and soundness conditions. The libraries of devices represented by causal theories are typically developed independently. Although each causal theory is written to maintain the consistency and soundness conditions, once they are merged to form a larger causal theory, these conditions are no longer guaranteed. The consistency condition and the soundness condition are necessary in order to support the unique cmi theorem. This section examines how these conditions can be maintained when merging two or more causal theories.

Violation of conditions at merging

The lack of extensibility in default logic has been pointed out in its early stage of development by Reiter[80] and some attempts have been made to overcome this limitation (for example [Reiter & Criscuolo 81]). The main focus of the problem here was that an arbitrary integration of new defaults leads to unwanted conclusions. The combination of causal theories suffers from the same problem since while each of the causal theories might be sound, but when two or more causal theories are merged, they might have contradictory defaults.

We first begin by summarising what conditions are violated in the case of causal theories. It is important to note that the conditions stated below are to support the *unique cmi theorem*, which allows only one *preferred* model.

7.7.1 Consistency condition

The consistency condition is defined as follows (p. 38):

> A causal theory Ψ is *consistent* if there do not exist two formulae in Ψ, whose antecedents are the same and the consequents are contradictory. For example, $\Phi \wedge \Theta \supset \Box(trm_{1a}, trm_{1b}, P_1)$ and $\Phi \wedge \Theta \supset \Box(trm_{2a}, trm_{2b}, P_2)$

where $eval(trm_{1a}) = eval(trm_{2a})$, $eval(trm_{1b}) = eval(trm_{2b})$, and predicates P_1 and P_2 are contradictory, are inconsistent.

This condition can be interpreted as "two inconsistent causal rules are not allowed". For example, consider following rules:

$$\Box(t,t,snow) \wedge \Diamond(t,t,slope) \supset \Box(t+1,t+1,go\_ski) \tag{7.5}$$

$$\Box(t,t,snow) \wedge \Diamond(t,t,slope) \supset \Box(t+1,t+1,\neg go\_ski) \tag{7.6}$$

Given condition TRUE(1,1,snow), rule (1) concludes that you can ski and rule (2) the opposite. If these rules were written within the same causal theory, it definitely is inconsistent and the knowledge engineer who made these rules is to blame; but this type of inconsistency can happen if the rules appear as the result of merging two causal theories which were developed by different persons. For example rule (1) might have been written as knowledge of an area where skiing is popular; on the other hand, rule (2) might be from an area where the snowfall often causes avalanches which endangers the activity on the slope.

The problem here, as is in most cases of inconsistent rules, is the implicit defaults in the rules. This kind of inconsistency can be fixed by the additional default conditions as follows:

$$\Box(t,t,snow) \wedge \Diamond(t,t,slope) \wedge \Diamond(t,t,\neg avalanche) \supset \Box(t+1,t+1,go\_ski) \tag{7.7}$$

$$\Box(t,t,snow) \wedge \Diamond(t,t,slope) \wedge \Diamond(t,t,avalanche) \supset \Box(t+1,t+1,\neg go\_ski) \tag{7.8}$$

This makes a clear distinction between the two rules as to what each rules assumed as default conditions. However this violates the soundness condition stated below.

7.7.2 Soundness condition

The problem caused by conflicting defaults is the violation of the soundness condition of a causal theory. The soundness condition is defined as follows (p. 38):

> A causal theory Ψ is *sound* if there do not appear in any antecedents of two formulae in Ψ, $\diamond(trm_{1a}, trm_{1b}, P_1)$ and $\diamond(trm_{2a}, trm_{2b}, P_2)$, where $eval(trm_{1a}) = eval(trm_{2a})$, $eval(trm_{1b}) = eval(trm_{2b})$, and predicates P_1 and P_2 are contradictory.

Since $\diamond$-conditions are considered to be the default conditions of the causal theory, it means that there should be no contradictory defaults within a causal theory. This prevents the application of inconsistent defaults in the course of inference.

The main research done in relation to the maintenance of such soundness is summarised below. This work was done in the context of nonmonotonic reasoning, and mainly deals with the "cyclic" relations caused by the addition of new rules.

Implicit ordering While Etherington and Reiter[83] suggests the semantics for the inheritance hierarchy, Touretzky[84] suggests that there is an implicit (partial) ordering between the default rules, by which the order of the application of the rules is determined. This ordering is determined by *influential distance*, which is based on the interaction between the prerequisites of the rules.

Scope Etherington *et al*[91] suggests the introduction of "scope" to restrict reasoning. It essentially forces circumscription to include the new predicate *Scope*. This restricts what individual to reason about.[3]

An example of using the scope in default logic looks like

$$\frac{\alpha(x) \wedge Scope(x) : \beta(x)}{\beta(x)}$$

thus specifying the scope of reasoning in its prerequisite. Note that the scope is always minimised so as to have similar features to circumscription.

This is a very straightforward technique but has drawbacks. First, how is the scope determined, and how does it change as the reasoning proceeds? Etherington claims

[3]This differentiates "scope" from *reference class* [Loui 88], which helps determine what default information to use when reasoning about an individual.

that he assumes the scope to be treated as the goal statement in problem solving, instead of being dynamically updated. This assumption is reasonable for the case of causal theories, where the scope of reasoning can be predetermined as an input from the user. The second issue and that limits its application to causal theories is that the computation becomes expensive as the number of individuals/terms defined in scope increases, and moreover, we must define another modelling procedure to cater for the scope.

Semi-normal defaults In order to enhance its expressiveness of his default logic, Reiter and Criscuolo[81] suggest semi-normal defaults to deal with interacting defaults. Essentially, semi normal defaults contain more information in their justifications. The main purpose of this formalism is to restrict the transitivity of the rules by explicitly including in their justification the conditions to prevent the rule to become valid.

An example of semi-normal default is as follows. Consider the sentences "Typically A's are B's and typically B's are C's. Typically A's are not C's." By using semi-normal defaults, we can represent this knowledge as

$$\frac{A(x) : B(x)}{B(x)}, \frac{B(x) : \neg A(x) \wedge C(x)}{C(x)},$$

$$\frac{A(x) : \neg C(x)}{\neg C(x)}$$

Notice that the second default was (before it was re-written into semi-normal form)

$$\frac{B(x) : C(x)}{C(x)}$$

As we can see, the normal default was re-written into semi normal default by including the $\neg A(x)$ in its justification, which prevents the transitivity $A(x) \rightarrow C(x)$.

The idea of this re-writing procedure is useful in the case with causal theories, but the discussion in the following section about the relationship between default logic and causal rules make it difficult to apply the rewrite rules directly to causal theories.

7.7.3 Enforcing the soundness condition

According to [Shoham 88], the default rules in default logic can be rewritten in terms of causal rules in the following way:

Default rule[4].

$$\frac{\alpha(x) : \beta(x)}{\gamma(x)} \tag{7.9}$$

to a causal rule

$$\Box\alpha(x) \wedge \Diamond\beta(x) \supset \Box\gamma(x), \tag{7.10}$$

This means that the semi normal defaults as suggested above can be translated into causal rules. There is, however, a restriction in the causal rule which makes this rewrite unreasonable. A normal default rule is of the form

$$\frac{\alpha(x) : \beta(x)}{\beta(x)} \tag{7.11}$$

which reads "if α and there is nothing inconsistent to assume β, then we can conclude β." Translating into a causal rule will be

$$\Box\alpha(t,x) \wedge \Diamond\beta(t,x) \supset \Box\beta(t+1,x) \tag{7.12}$$

which has a slightly different reading from before, because of an "advance" in time in the consequent. The reason why semi normal defaults cannot be directly applied in the case of causal rules is the same.

The best way to maintain the soundness is, then, to "promote" the unsound default conditions to necessary conditions. That is, suppose there are two rules with inconsistent default conditions

$$\Box(t,t,p_a) \wedge \Diamond(t,t,d_1) \wedge \Diamond(t,t,d_2) \supset \Box(t+1,t+1,q_a) \tag{7.13}$$

$$\Box(t,t,p_b) \wedge \Diamond(t,t,\neg d_1) \wedge \Diamond(t,t,d_3) \supset \Box(t+1,t+1,q_b). \tag{7.14}$$

We cannot assume conflicting defaults at the same time, so we must choose one of them, say d_1. Then assuming $\neg d_1$ is no longer valid as a default condition. We now "qualify" it as a necessary condition by rewriting rule 7.14 as

$$\Box(t,t,p_b) \wedge \Box(t,t,\neg d_1) \wedge \Diamond(t,t,d_3) \supset \Box(t+1,t+1,q_b). \tag{7.15}$$

[4] $\alpha(x), \beta(x), \gamma(x)$ are well-formed formulae called, respectively, the *prerequisite*, the *justification* and the *consequent* of the default

In the example in section 2, if we choose as the default that there is no chance of avalanche, the rule 7.8 is then rewritten to be

$$\Box(t,t,snow) \wedge \Box(t,t,avalanche) \wedge \Diamond(t,t,slope) \supset \Box(t+1,t+1,\neg go\_ski) \quad (7.16)$$

and by doing this we do not lose any information or expressiveness of the rules. Table 7.1 shows what can be concluded from each condition; notice that the conclusions we lose are undesirable ones.

Table 7.1: Conclusions for various conditions.

conditions	7.13 and 7.14	7.13 and 7.15
p_a	q_a	q_a
$p_a \wedge d_1$	q_a	q_a
$p_a \wedge \neg d_1$	—	—
p_b	q_b	—
$p_b \wedge d_1$	—	—
$p_b \wedge \neg d_1$	q_b	q_b
$p_a \wedge p_b$	$q_a \wedge q_b$	q_a
$p_a \wedge p_b \wedge d_1$	q_a	q_a
$p_a \wedge p_b \wedge \neg d_1$	q_b	q_b

7.8 Optimisation measure

Generally, the modification process involves various choice points. There are likely to be more than one possible modifications to the design and we must therefore select a modification from a set of candidates. In this context, such selection criteria are treated as optimisation measures. Optimisation is essential at this stage not only to obtain the most preferred solutions but to reduce the complexity of generation of design. In most cases, the suggestions for legally correct modifications are produced purely syntactically; optimisation measures guide the selection of the preferred modification by incorporating interpretive knowledge [Coyne *et al* 90, p.379].

Below are two of the optimisation measures applied in the general design process.

- *Numeric measures*—Optimisation of certain parameters is usually based on certain optimisation functions. Examples of such functions are linear programming

and hill-climbing techniques which are used in the field of operations research (OR) and various engineering domains. Parameters are assigned by minimising or maximising particular values of functions.

- *Purpose oriented measures*—Design can be optimised to suit particular purpose. The ways to achieve this can be by assigning an optimisation function if there is some well-defined formula, or by simply comparing the candidates in some semantical way. For instance, when designing a bridge, there could be two candidates for its style, say suspension and truss, there can be a preference for the suspension bridge from aesthetical point of view [Smith *et al* 91].

Clearly, these measures are heuristics that are considered to provide the most preferred choice according to some particular purpose. Sometimes the optimisation itself is a solution to a design problem if the problem is mathematically well formed, in terms of criteria, constraints and decision variable. Design problems can be underconstrained or overconstrained but preferences are useful in most cases; it can be used to guide search if underconstrained as in many search heuristics, and if the design is overconstrained, preferences can guide the search for the best tradeoff [Smith 92].

In the proposed approach, the design is optimised to have as small number of changes as possible. This guideline is followed by the preference to potential persistence backwards to control the abductive process (Section 7.5). That is, by assuming that things should change as little as possible, the preferred design is that which has least number of occurrence of events when simulated. This optimisation measure is likely to achieve the following features:

- **Small number of components.** Since the least number of changes is preferred, the components involved should be smaller. This does not guarantee the *minimal* number of components, but it is generally true that designs with smaller number of components are preferred, for cost performance and the reduction of possible occurrence of faults.

- **Small number of connections.** As was the case with the number of components, the preference for the least changes reduces the number of connections

between the components, thus simplifying the interfaces between them.

- **Conservative modifications.** Since the abduction framework is applied to the existing causal theory, existing components and relations are used as much as possible. For example, if there is a need to establish a causal relation between A and B ("if A then B"), and if there is a device which renders the causal relation between C and B ("if C then B"), it suggests a device that obtains C given A, instead of trying to create something that directly obtains B from A.

In addition to the built in preference criteria described above, we can have domain dependent heuristics to choose a candidate among others. Such interpretive semantic knowledge complements the application of logic CI and its abduction framework which in itself does not carry design knowledge. These heuristics are applicable when the selection is among those of the same type and/or abstraction. An example of such a case is the selection of the type of flip flop circuits where the external behaviour of the circuits are all same and therefore the choice of which type of circuit to use to compose a device should not affect the overall behaviour.

7.9 Theory refinement and device compilation

After the design of a device is complete—desirable overall behaviour has been achieved for all scenarios considered –it would be useful to have it stored for re use. As long as the device is used to provide a certain function as it is, there is no need to decompose it again to be re-designed. It only needs to be decomposed when the only option is to modify the device to suit an additional purpose. This section describes briefly a way to compile a device which has been designed and verified.

7.9.1 Theory refinement

Once the behaviour of a device is modelled and proved to be plausible, it is useful to have it stored in the library for later use. For example, once the causal rule description of a solar cell is considered to be reasonable, it will be listed along with other devices.

The rules can be made abstract if there is no secondary effect during their operation. Those events which would affect the environment (e.g. the increase in temperature, turning the light on, etc.) are called *external* events, as opposed to *internal* events, which occur only within a device (e.g. change in the flow within closed pipe circuit, increase of pressure within a container, etc.). Given an input event p_{input} and the output p_{output}, and the chains of events $q_1 ... q_n$ between them, i.e.

$$p_{input} — q_1 — \cdots — q_n — p_{output},$$

we need only to keep explicit those external events among q_i which would affect other histories. In this causal chain, there are at most $n+1$ causal rules involved. Assuming that the external events are q_4 and q_{11}, then the set of rules can be reduced to

$$\Box(t, t, p_{input}) \wedge \Theta_1 \supset \Box(t+1, t+1, q_4)$$
$$\Box(t, t, q_4) \wedge \Theta_2 \supset \Box(t+1, t+1, q_{11})$$
$$\Box(t, t, q_{11}) \wedge \Theta_3 \supset \Box(t+1, t+1, p_{output}).$$

These rules now represent the functions of the device. When describing a complicated circuit which has been designed, we no longer need to talk about every component and connection which constitutes the circuit, but only about the inputs and the outputs of the circuit. These can be considered to be a high level description of the underlying causal theory and are compiled in the component library.

7.9.2 Component library

The components available to the domain need to be stored in a library, with their contribution to the causal theories. What is stored in the *component library* is the knowledge about what causal rules are attributed to particular components. This provides a higher-level description of the design, which is by identifying the necessary components and their relations to each other.

7.10 Summary

This chapter described the notion of design modification via the construction of a causal theory which exhibits the desired behaviour. Notable points are:

- The modification of design is performed given the results obtained from the verification phase, and involves abduction of causal rules which exhibit the missing causal relations.
- Since components and connections are represented by causal theories, the addition/modification/removal of components is performed by the same operations to the corresponding causal theories.
- Such operations need to guarantee the soundness and consistency conditions of the produced causal theory.
- Techniques based on bidirectional sweeping are used to guide the choice of modifications which prefer the smallest number of changes.

The results of modification are fed back to the verification phase to form the 'generation' part of generate-and-test design cycle.

Chapter 8

Design problems

This chapter presents some sample problems which illustrate the behaviour-oriented design process described in the previous chapters. Although the scale of the problems here are relatively small, our aim is to demonstrate how the problems are formulated and illustrate the procedures of the proposed method. We are far from claiming that this technique is better than whatever conventional design methods have been developed in the particular domain; each domain normally has its most appropriate design method which has been optimised over many years of experience and research. The contribution from the proposed framework is to show that certain aspects of design can be modelled and represented from an alternative perspective.

8.1 Properties of applicable domains

Both of the sample problems presented here are in the logic circuits domain, where the functionality of components are well defined and relatively restricted. Such features provide a good basis to test the framework of behaviour-oriented design and how each phase in design works.

The problem domains where the proposed method is considered to be applicable are those with the following properties:

- Events can be represented as discrete, grounded and finite states.

- Estimated combinatorics are relatively small, e.g., the number of components, possible connections between them, possible states, etc. are small enough to be controlled interactively.
- Designs are heavily dependent upon the behaviour, or temporal aspects, of the device.

Examples of the domains which might have these properties include digital electronic circuits, process engineering (e.g. processing plants), machine assembly and a range of configuration problems.

8.2 Implementation issues

The implementation of the system consists of five modules; the cmi model construction module, discrepancy detection module, rule construction module and rule proving module.[1] The function of each module is as follows:

Model construction module constructs the cmi model from given causal theory.

Discrepancy detection module compares the cmi model and the behavioural specification, and identifies the earliest discrepancy.

Rule construction module suggests a new rule which fixes the discrepancy.

Rule proving module performs rewriting operations (see Section 7.6.1) and suggests a rule to assert.

At this stage, there is an interaction between the user and the system. The user is asked whether the rule suggested is reasonable. If not, the system will suggest an alternative, and this process continues until the user finds the suggested rule reasonable. Since the user interface is not considered to be an issue here, minimal automation is provided, and rule assertions are conducted manually in this implementation. The user interface can be improved, for instance, in the way that it supports the interactive addition and

[1] The main parts of the code is included in Appendix C.4.

retraction of rules into the causal theory, labelled with dependencies as to which rules were modified for which scenario, to enable backtracking in the modification process.

8.3 Sample problems

The sample problems presented here in the logic circuit domain have the above properties and serve as typical examples. Here we present two examples, the configuration of set-reset flip-flop circuit, and the simulation of JK flip-flop circuit.

8.3.1 Set-Reset flip-flop circuit

A logic circuit in which the output depends on the temporal order of the application of inputs is called a *sequential circuit*. In sequential circuits, a particular sequence of inputs must be provided in order to obtain required outputs.

The set-reset flip-flop circuit (SRFF circuit) is a very simple sequential logic circuit which are used in many sequential systems. S and R are two inputs to the circuit, and Q and Q' being its outputs.

The behavioural requirement for the SRFF circuit is described as follows (extracted from [Gibson 83, page44]):

(a) S and R are normally held at 0 and the outputs remain constant in either one of the $Q = \overline{Q'}$ states.

(b) An input sequence of 0 to 1 then back to 0 at the S input will ensure that $Q = 1$ and $Q' = 0$.

(c) A similar $0 - 1 - 0$ input sequence at the R input ensures that $Q = 0$ and $Q' = 1$.

(d) In normal circuit design the input condition $S = R = 1$ should not be allowed.

(e) If power is connected to the circuit with $S = R = 0$, the circuit will take either one of the states $Q = \overline{Q'}$.

where $\overline{Q}$ denotes the inverse of Q, or *not* Q, i.e., if $Q = 1$ then $\overline{Q} = 0$ and vice-versa.

To simplify the task to a certain extent, what we attempt to design here is $\overline{\text{SR}}$ flip-flop circuit ($\overline{\text{SR}}$FF circuit) which is a variant of SRFF circuit. $\overline{\text{SR}}$FF circuit exhibits behaviours similar to SRFF circuits, but all the inputs are inverse of that of SRFF. Therefore, the behavioural requirements for $\overline{\text{SR}}$FF circuit is described as follows:

1. $\overline{S}$ and $\overline{R}$ are normally held at 1 and the outputs remain constant in either one of the $Q = \overline{Q'}$ states.
2. An input sequence of 1 to 0 then back to 1 at the $\overline{S}$ input will ensure that $Q = 1$ and $Q' = 0$.
3. A similar $1-0-1$ input sequence at the $\overline{R}$ input ensures that $Q = 0$ and $Q' = 1$.
4. In normal circuit design the input condition $\overline{S} = \overline{R} = 0$ should not be allowed.
5. If power is connected to the circuit with $\overline{S} = \overline{R} = 1$, the circuit will take either one of the states $Q = \overline{Q'}$.

Among the five items listed above, items 1 and 5 provide static constraints of the circuit when both $\overline{S}$ and $\overline{R}$ inputs are held at 1. Item 4 also describes the condition which is not allowed. This leaves us with items 2 and 3 from which we can obtain following two scenarios:

$$S_1 = \{\Box(1,3,s(1)), \Box(4,6,s(0)), \Box(7,9,s(1))\}$$
$$S_2 = \{\Box(1,3,r(1)), \Box(4,6,r(0)), \Box(7,9,r(1))\}.$$

where s and r describe the values of $\overline{S}$ and $\overline{R}$ respectively. It assumes that each state for $\overline{S}$ and $\overline{R}$ would last three time steps. By taking the constraint imposed by 4 into consideration, we can amend these scenarios as follows:

$$S_1 = \{\Box(1,3,s(1)), \Box(4,6,s(0)), \Box(7,9,s(1)), \Box(1,9,r(1))\}$$
$$S_2 = \{\Box(1,3,r(1)), \Box(4,6,r(0)), \Box(7,9,r(1)), \Box(1,9,s(1))\}.$$

These describe the input sequence of $1-0-1$ for $\overline{S}$ and $\overline{R}$ respectively without violating the condition 4. From the behavioural requirements, behavioural specifications for each scenario are described as follows.

$$BS_1 = \{(1,3,s(1)),(4,6,s(0)),(7,9,s(1)),(1,9,r(1)),(10,\infty,q1(1)),(10,\infty,q2(0))\}$$
$$BS_2 = \{(1,3,r(1)),(4,6,r(0)),(7,9,r(1)),(1,9,s(1)),(10,\infty,q1(0)),(10,\infty,q2(1))\}.$$

where $q1$ and $q2$ describe the values of Q and Q' respectively. The environment rules include the causal rules for the logic gates. For example, the function of a NAND gate is represented by the causal rules

1. $\Box(t,t,input(X,a,1)) \wedge \Box(t,t,input(X,b,1)) \supset \Box(t+1,t+1,output(X,0))$
2. $\Box(t,t,input(X,a,0)) \wedge \Diamond(t,t,input(X,b,N)) \supset \Box(t+1,t+1,output(X,1))$
3. $\Box(t,t,input(X,b,0)) \wedge \Diamond(t,t,input(X,a,N)) \supset \Box(t+1,t+1,output(X,1))$.

where a and b represent the two inputs into a NAND gate X, and N represents the values of either 0 or 1.

Other gates such as AND gates, OR gates, NOT gates etc. can be described by a set of causal rules as NAND gates.

In this example we assume as a prototype design, NAND gates without any connection between them. However, to simplify the problem, we also assume that one of the inputs of a NAND gate (*nand1*) is connected to $\overline{S}$ signal, and one of the inputs of another NAND gate (*nand2*) is connected to $\overline{R}$ signal. In addition to these, we assume as the prototype design, that Q is connected to the output of *nand1* and Q' to the output of *nand2*. These provide the following causal rules, and the configuration of the prototype is illustrated in Figure 8.1(a).

4. $\Box(t,t,s(N)) \supset \Box(t+1,t+1,input(nand1,a,N))$
5. $\Box(t,t,r(N)) \supset \Box(t+1,t+1,input(nand2,b,N))$
6. $\Box(t,t,output(nand1,N)) \supset \Box(t+1,t+1,q1(N))$
7. $\Box(t,t,output(nand2,N)) \supset \Box(t+1,t+1,q2(N))$

The causal theory of the system is then constructed by collecting the causal rules and the boundary conditions for the environment, scenario and design. In this case, causal

rules 1 to 7 along with S_1 provide the causal theory for the first scenario. The soundness and consistency of the causal theory is then checked and dealt with whenever these conditions are violated. The cmi model computed for the causal theory above is:

time point	events
1	$r(1), s(1)$
2	$r(1), s(1), input(nand1, a, 1), input(nand2, b, 1)$
3	$r(1), s(1), input(nand1, a, 1), input(nand2, b, 1)$
4	$r(1), s(0), input(nand1, a, 1), input(nand2, b, 1)$
5	$r(1), s(0), input(nand1, a, 0), input(nand2, b, 1)$
6	$r(1), s(0), output(nand1, 1), input(nand1, a, 0), input(nand2, b, 1)$
7	$q1(1), r(1), s(1), output(nand1, 1), input(nand1, a, 0), input(nand2, b, 1)$
8	$q1(1), r(1), s(1), output(nand1, 1), input(nand1, a, 1), input(nand2, b, 1)$
9	$q1(1), r(1), s(1), output(nand1, 1), input(nand1, a, 1), input(nand2, b, 1)$
10	$q1(1), input(nand1, a, 1), input(nand2, b, 1)$

The verification of this design by comparing this with the behavioural specification BS_1, the system detects that the event $q2(0)$ is missing ('insufficient'). In order to cater for this, the system suggests a number of possible additions to the causal theory, some of them being asserting one of the following rules.

$$\Box(t, t, q1(1)) \supset \Box(t+1, t+1, q2(0))$$
$$\Box(t, t, r(1)) \supset \Box(t+1, t+1, input(nand2, b, 1))$$
$$\Box(t, t, s(1)) \supset \Box(t+1, t+1, input(nand2, a, 1))$$
$$\ldots$$

However, many of these candidates can be eliminated if we have the design knowledge such as the output Q cannot be an input to anything, inputs $\overline{S}$ and $\overline{R}$ are connected to only one terminal, a NAND gate cannot have two inputs to the same terminal, etc., the candidates can be restricted. Among them we can find

$$\Box(t, t, output(nand1, 1)) \supset \Box(t+1, t+1, input(nand2, a, 1))$$

which suggests the connection added in Figure 8.1(b).[2] Asserting this causal rule and

[2]Since this causal rule only suggests the propagation of signal 1 between these terminals, it is in theory yet premature to add this feedback connection. We provide this to illustrate the connection for time being.

reconstructing the cmi model results in 'no discrepancy' which means that Scenario S_1 has been fulfilled.

We conduct the same procedure for scenario S_2, which identifies that the event $q1(0)$ is missing after the input sequence. Among other candidates which is analogous to those for scenario S_2, it suggests to assert

$$\Box(t, t, output(nand2, 1)) \supset \Box(t+1, t+1, input(nand1, b, 1))$$

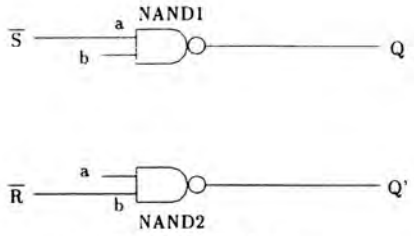
which provides a feedback connection from the output of NAND2 to an input to NAND1.

By comparing the predicted behaviour and the behavioural specification, the necessary connections between the terminals are found. Figure 8.1(c) shows the connections constructed under scenario S_2 and the behavioural specification BS_2.

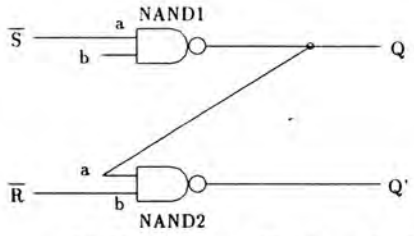
This sample session is described in Appendix B.3. The session relies heavily on the interaction between the user and the system; the system merely suggests rules to be added and the user decides which rules to choose and which to reject. In the current implementation, there is no facility to support backtracking to previous rule sets prior to any rule assertion.

It is important to note that Figure 8.1 is one of several solutions obtainable by the system. Some solutions involve multiple assertions of NOT gates (a sequential pair of these is equivalent to a mere connection), or redundant NAND gates. There is currently no valid optimisation procedure in the process, but there are some heuristics which prevent obviously unreasonable combinations. One plausible optimisation is to prefer those which introduce the least number of components. In this sense, the solution given above is optimal.

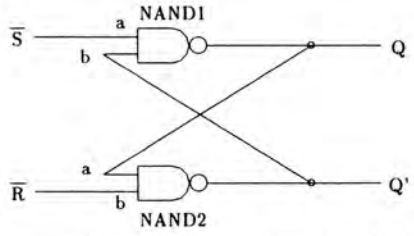
The significance of this example is that although it is simple, the design reasonably suggests the feedback links between the outputs and inputs of NAND gates.



(a) Initial configuration



(b) Wiring after modification under Scenario1



(c) Wiring after modification under Scenario2

Figure 8.1: Configuration of $\overline{SR}$ flip-flop circuit: (a) describes the prototype which is the initial configuration. (b) shows the connection added after examining scenario S_1, and (c) is the configuration after scenario S_2 was examined.

8.3.2 Simulation of asynchronous circuits

One of the problems in the domain of logic circuits, which involves more temporal constraints, is that it requires the representation of delays. An example is an asynchronous binary counter such as illustrated in Figure 8.2.

It is composed of three JK flip-flops (JKFFs); the JKFF is one of the most complicated flip flops and is used in many sequential logic circuits [Gibson 83]. It has two control inputs J and K, and a clock input. Its two outputs are Q and $\overline{Q}$. A JKFF circuit can be constructed in several ways, one of which is a master-slave JKFF described in Figure 8.3. Table 8.3.2 describes the action table of a JKFF.

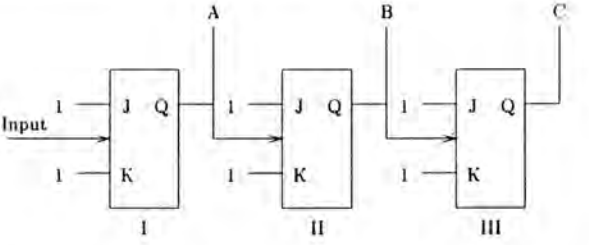
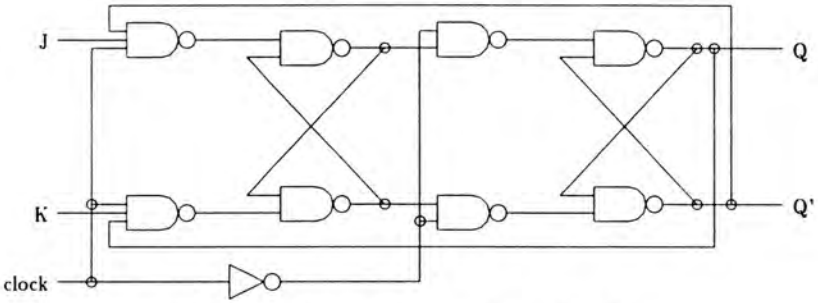


Figure 8.2: Asynchronous divide-by-eight counter: the rectangles (I,II,III) represent JK flip-flops, and A,B,C the output terminals.

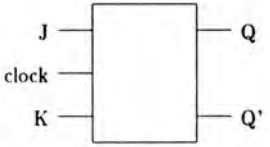
Table 8.1: The action table of a JK flip-flop. A clock pulse is equivalent to a sequence of 0-1-0 input.

J	K	State after next clock pulse
0	0	No change in the value of Q
1	0	Q is set to 1
0	1	Q is set to 0
1	1	Q toggles from 0 to 1, or from 1 to 0

We can describe the behaviour of JKFF without referring to its structure (e.g. a set of NAND gates) and treat it as a component with well-defined behaviour. The set of



The master-slave JK flip-flop circuit with NAND gates



JK flip-flop as a component

Figure 8.3: The master-slave JK flip-flop circuit.

causal rules below describes the behaviour of a JKFF.

1. $\Box(t,t,jkin(X,0,0)) \wedge \Box(t,t,jkout(X,Q1,Q2)) \wedge \Box(t,t,clock(X))$
   $\supset$ POTEN$(t+1,\infty,\mathrm{p}\text{-}jkout(X,Q1,Q2))$
2. $\Box(t,t,jkin(X,1,0)) \wedge \Box(t,t,clock(X)) \supset$ POTEN$(t+1,\infty,\mathrm{p}\text{-}jkout(X,1,0))$
3. $\Box(t,t,jkin(X,0,1)) \wedge \Box(t,t,clock(X)) \supset$ POTEN$(t+1,\infty,\mathrm{p}\text{-}jkout(X,0,1))$
4. $\Box(t,t,jkin(X,1,1)) \wedge \Box(t,t,jkout(X,Q1,Q2)) \wedge \Box(t,t,clock(X))$
   $\supset$ POTEN$(t+1,\infty,\mathrm{p}\text{-}jkout(X,Q2,Q1))$.

where *jkin(X,J,K)* represents the values of the inputs J and K for a JKFF X, *jkout(X,Q1,Q2)* represents the output values Q=*Q1* and Q'=*Q2* for a JKFF X, and *clock(X)* represents that there is a clock pulse for a JKFF X.

We can now construct the binary counter illustrated in Figure 8.2. In this device, the output value of each JK flip flop is the input of the next one and the change in the outputs propagates from I to III. Since each JK flip-flop must wait for the change in the previous one, there are delays between the changes in the outputs A,B,C which make it asynchronous.

In this example, the delay between the change of the value in each output and the input to the next flip flop plays the important part in the behaviour of the system. To illustrate this, the causal rules which describes the clock inputs to JKFFs are described as follows:

5. $\Box(t,t,clock(input)) \supset \Box(t+1,t+1,clock(jkI))$
6. $\Box(t,t,jkout(jkI,1,0)) \wedge \Box(t+1,t+1,jkout(jkI,0,1)) \supset \Box(t+2,t+2,clock(jkII))$
7. $\Box(t,t,jkout(jkII,1,0)) \wedge \Box(t+1,t+1,jkout(jkII,0,1)) \supset \Box(t+2,t+2,clock(jkIII))$

Here, *clock(input)* is the external clock input, and *clock(jkI), clock(jkII), clock(jkIII)* denote the clock inputs to JKFFs *jkI, jkII* and *jkIII*, respectively. Rule 6 shows that the clock pulse into jkII is the result of transient of output Q of jkI.

The temporal ordering causes delays between the changes in the output of I and the input to II. Such representation of delays produces an accurate model for the behaviour of the counter. In this case, we can represent each JK flip-flop by the causal relation between its inputs and outputs; in other words, although it consists of an SR flip-flop and some NAND gates, it is not necessary to describe them in terms of subcomponents. Once a device is demonstrated to show a certain function, we can compile it as an available device in the environment. The output sequence of the circuit is given in Table 8.2 and it shows that there are transients between the upward changes in the value.

Table 8.2: The output sequence of the binary counter.

t_1	C	B	A	
t_2	0	0	0	initial state
t_3	0	0	1	
t_4	0	0	0	transient state
t_5	0	1	0	
t_6	0	1	1	
t_7	0	1	0	transient state
t_8	0	0	0	transient state
t_9	1	0	0	
	...	...		

The significance of this example is that the representation with causal rules is capable of producing a precise simulation of the system. This feature becomes very important when we are concerned not only by 'what' is actually achieved, but 'how' it is achieved.

8.4 Evaluation of the method

The sample problems described in the previous sections suggests that the behaviour-oriented approach to design can indeed help construct and verify plausible configurations of sequential logic circuits. We shall now take a look at some of the issues that arise from its process.

8.4.1 Computational efficiency

As has been seen in the SRFF circuit example, a number of candidates for modification of design can be suggested. The number of such candidates were quite large considering the size of the problem, and once the number of components, domain rules, and events appearing in behavioural specifications scales up, there would be too many of such candidates to manage. The sources of this complexity are the abduction process which takes place in suggesting the possible events to obtain necessary events, and the number of components which can serve as potential candidates. This complexity is a major obstacle to practical usage of this method. The selection of candidates can be partly dealt with by incorporating specific design knowledge which may be provided as heuristics.

While this could prove to be a major drawback of this method, the main purpose of this approach is the investigation of how behaviour-oriented approach can be performed through causal reasoning. Also, this method is not intended for automated design, but rather aimed at providing guidance through interaction between the system and the user. In the longer term, by incorporating it into a more elaborate design framework, we might be able to make use of other design knowledge and/or user interaction to control the process.

8.4.2 The validity of cmi model as the prediction

The unique cmi model shows that there is only one most preferable model for a causal theory [Shoham 88]. This might be too restrictive if we consider the possibility of having multiple possible behaviours from a design. To focus on the most likely outcome would raise the danger of overlooking a non-deterministic aspect of the design. It might be more informative if we could maintain all possible behaviours in order to evaluate the design.

Focusing on the unique model, however, does not exclude other possible behaviours. It is merely suggested as 'the most likely behaviour' under 'given conditions and (default) assumptions'. Typically, a causal theory would split into different histories depending

on whether the default is overridden, since there is no need to state the default conditions in a causal rule when they do not affect the causal outcomes. Whenever the predicted behaviour contradicts the expected behaviour, there is always a possibility that the default assumed is not right. We can check this by listing out the default conditions used in the modelling process, and if there is any 'wrong' default, it is easily overridden by adding a boundary condition. In most cases, multiple worlds occur due to lack of information about specific conditions that distinguish the worlds.

There is another case where there arises more than one behaviour. When a variable is universally quantified and there is no need to assign any value to it, there are as many possible worlds as the number of members in the set of individuals semantically assigned to the variable. A typical example of this case is the 'don't care' state in the logic gates. Since there are no restrictions to the value of the variable, assuming the 'most likely' value to it seems to be plausible.

The reason why we think unique model is appropriate is that it gives flavour of how a device would behave by default. If the designer is satisfied with the outcome, maybe using the default is good enough; if not, s/he can find out which default to override. In case of multiple worlds, we can see what 'can' happen, but sometimes the number is exceedingly high to manage. The process of choosing the 'right' one would nevertheless be the same.[3]

8.4.3 Conflicting specification

The evaluation-modification cycle of the design process does not always work, especially when there is a conflict in the initial specification. Although the proof process in the modification stage is aimed to make an abstract causal rule to be a feasible one, it is not always guaranteed to be the case. Since it is based on a mechanical application of operations, sometimes the 'feasible' rules may still be counterintuitive or simply impossible.

[3] In this context, we are assuming that a device would behave uniquely under a given condition. There are cases when a device should have multiple behaviours, but we assume this happens only when the design is under-constrained.

For example, the causal rule might give a description of a device which reads "when the power of the magnetic field increases, the mass decreases", or "when a given object is travelling at the speed of light, it accelerates."[4] When such impossible causations are derived as the most plausible among other alternatives, there is something wrong with the behavioural specification.

In such cases, there will be no alternative but to modify the specification. Since there will be some index to the boundary conditions that are responsible for the undesirable events after the failure of the proof, it is possible to figure out the conditions under conflict. Automatic resolution of these conflicts is not in the scope of the current research, but it should be possible to support the designer by indicating the necessary conditions to obtain the desired behaviour.

8.4.4 Choice of temporal reasoning formalism

Although the proposed framework for behaviour-oriented design is constructed using Shoham's temporal logic, the basic idea should be portable across other temporal reasoning formalisms. Some guidelines for choosing a temporal reasoning formalism are as listed below:

- It should be able to represent sequences of events to formulate a behavioural specification.
- It should be able to deal with both prediction and reconstruction tasks.
- Preference criteria are either built into the framework, or can be specified.

This last item in the list may seem to be restrictive; but it is an important feature when one is dealing with problems which are underspecified or ill-structured as design. It is possible to apply a temporal reasoning framework that maintains all possible outcomes of prediction, but it is quite difficult to manage if there are too many of these outcomes as a result of a loosely specified problem.

[4]The former is the case when the causal connection is not obvious or non-empirical, but *may be* possible with additional knowledge about physics. The latter is simply impossible to have such an effect in real physics.

The central issue is that regardless of what temporal reasoning formalism is chosen we must find an appropriate way to represent causal knowledge to provide the elements of behaviour-oriented design. And the capability of the formalism to capture the design phases of simulation, verification and modification is an essential feature for representation and reasoning about behaviour in design.

8.5 Summary

The essence of the approach illustrated in this chapter is to treat the functionality of design components in terms of causality. Causal relations among events produce a chronological sequence of events (or causal chains), and along with the causal rules describing the design components, we can specify the behaviour of the device. The causal rules are manipulated according to Shoham's temporal logic which incorporates the nonmonotonic aspect of the prediction task. The formulation is based on a well-defined theory, with clearly defined theoretical justification.

The advantages of this approach are

- the behavioural specification of devices allows more intuitive descriptions of the functionality of the system, especially in dynamic systems where the devices interact with the environment, and
- it utilises temporal constraints for design evaluation and modification, which enables the representation of delays and relative timings in the occurrence of the events. Without temporal logic, we must rely on the implicit ordering of events and it is difficult to represent the relative ordering of events.

On the other hand, the main drawback in the design stage of this approach is its complexity. There are typically a number of candidates for which rule to choose for the abduction process, and the combinatorics become more complex as the number of such rules and the number of conditions in their premise become greater. At the moment, some primitive heuristics are used to reduce the search space, and some choices are reserved for the human user to decide. This permits the users to incorporate

their preference, but as the number of rules increase in a more complex domain, it is necessary to introduce more elaborate optimisation procedures to guide search.

Chapter 9

Conclusion and future perspectives

This dissertation consists of two parts. The first part develops a formalism to deal with reasoning about change in prediction and reconstruction tasks. It is based on the language CI introduced by Shoham. The notion of chronological minimisation is applied to reasoning backwards, and bidirectional sweeping into both the past and the future provides some solutions for interpolation problems. The second part of this dissertation suggests an application of the formalism to design problems. By focusing on how devices work, I put forward the notion of behaviour-oriented design which has the aim of achieving stipulated behaviours of devices and their interactions with their surroundings. Suggestions for future work follow the description of the contributions of this work and the conclusions.

9.1 Summary and Contributions

To summarise in a single paragraph, this dissertation has presented the following. Chapter 2 discussed causation and causal reasoning and related technicalities. Chapter 3 examined and implemented Shoham's logic CI and extended it to the first-order case to enhance expressiveness. Chapter 4 proposed the application of chronological minimisation backwards in time to treat reconstruction tasks, and presented bidirectional sweeping to deal with interpolation tasks. The feature of the role of causal

reasoning was discussed in Chapter 5. Chapter 6 introduced the concept of behaviour-oriented design and proposed a framework for this paradigm. Chapter 7 described design verification by comparing specified and predicted behaviours of a design. Some sample problems were presented in Chapter 8 to illustrate the proposed design support method.

The contributions of this work are summarised as follows:

An extension of the logic CI to a restricted first order case. In this work, the logic CI developed by Shoham was extended from the propositional case to a restricted first-order case. This enhanced the expressiveness of the language and provided a basis for its wider application.

Representation of practical problems with logic CI. The logic CI was implemented and applied to problems of a greater size. A qualitative reasoning problem was represented in CI, thus showing that CI is capable of representing and reasoning with such problems.

Development of an abduction framework for logic CI to deal with reconstruction tasks. A framework to perform abduction in logic CI was devised to deal with reconstruction problems. It employed the notion of persistence backwards into time to prefer sets which have the smallest number of deflections, and those with a minimal number of assumptions. In addition, the notion of forward and backward sweeping was presented to deal with interpolation problems. It demonstrated that the inconsistencies between the prediction and the reconstruction provide pointers to the missing events which constitute the causal progression of events.

Development of a framework for behaviour-oriented design. The notion of behaviour-oriented design, which focuses on the behaviour of a device and its surroundings, was suggested and provided a framework for the representation and reasoning using logic CI and its abduction framework. The advantage of behaviour-oriented design

is that the specification is described in terms of how a device should perform in given environments, which was not explicit in conventional representations for design.

9.2 Conclusions

This study suggests the following conclusions.

1. Extending Shoham's logic CI to the restricted first-order case, adding more expressiveness to the language, and its efficient implementation make it capable of computing the unique cmi model for larger problems than those demonstrated by Shoham.

2. Application of preference to persistence of events backwards provides reasonable solutions for reconstruction problems. The abduction framework developed for the logic CI provided the mechanism for computing such sets. The main advantage of this abduction framework is that it is applied to the same set of causal theories used for prediction in CI.

3. Interpolation problems can be treated by sweeping forwards and backwards by constructing the cmi model and bci sets. The identification of events which bring the cmi model and bci set closer provided plausible solutions to some simple benchmark problems.

4. The framework of behaviour-oriented design can focus on the way a device should perform over time, providing an additional perspective of formulating design problems. It suggests that causal reasoning can play an important role in this framework, by constructing behaviours and verifying a design on the basis of its behaviour.

9.3 Further work

A number of further enhancements are envisaged for this system, mainly to extend its capabilities as a practical design tool.

9.3.1 Alternative preference criteria

The use of logic CI and its abduction framework introduces the preference criterion of chronological minimisation to their prediction and reconstruction mechanisms. The application of this type of minimisation has been widely accepted in the prediction tasks [Sandewall 92a], there is yet more to be investigated in its application to reasoning about the past in various frameworks. Works in abduction suggest the notion of *minimal explanations* [Reggia *et al* 83] [Cox & Pietrzykowski 86] [Levesque 89], but such explanations are not always *chronologically* minimal. A minimal explanation merely minimises the number of assumptions which are introduced in abduction and this mirrors the global minimisation [McCarthy 86] with which the Yale Shooting Problem could not be solved.

Although what we have seen in this thesis was an attempt to apply the persistence criterion for both forwards and backwards which provided reasonable solutions to reconstruction and interpolation problems, this may not be the most appropriate basis for preference. This requires further investigation and experimentation.

Maintaining multiple worlds

The preference criterion enables the construction of the unique cmi model in the logic CI, and this is considered to be one of its advantageous features. At the same time, however, it poses a limitation of the logic CI in that it does not handle multiple outcomes. This results in not being able to express causal knowledge which involves disjunction in its conclusion such as

$$\Box(t,t,a) \wedge \Diamond(t,t,b) \supset \Box(t+1,t+1,c_1) \vee \Box(t+1,t+1,c_2)$$

where c_1 and c_2 are two possible, but mutually exclusive conclusions.

If we were to modify the logic to allow such causal rules, one option is to maintain a set of possible worlds and compute the unique model for each world; i.e., we compute the unique model for each choice of disjuncts, retaining the information about what choice of disjuncts was made and when. The predicted outcome will then itself be

conditional, e.g., 'if we assume c_1 to occur as the consequence of a and b, then we can conclude d_1; else if we assume c_2 instead of c_1, then d_2 is predicted.' This sounds like a plausible prediction.

Furthermore, if multiple worlds are maintained, we can reproduce the results of qualitative simulation by this causal reasoning framework. We have attempted this in Chapter 4 and produced a satisfactory result with unique models. However, some may argue that application of such a preference criterion makes qualitative simulation lose the beauty of producing all possible state transitions in its envisionment. Keeping multiple worlds will retain this feature.

We must not forget that by maintaining multiple worlds, we will introduce a great increase in computational complexity. One of the earlier criticisms of qualitative reasoning and the issue where much effort continues to be put into this field is the control of so many possible outcomes. At the same time, which style is better, computing a unique model or maintaining possible worlds, depends on what you are trying to solve. As a research issue, however, it is interesting to integrate the chronological minimisation approach with the idea of probabilistic temporal reasoning [Dean & Kanazawa 88].

Nontemporal conditions

Causal rules in CI do not allow cause and effect to be simultaneous, i.e., cause and effect sharing a same time point. The issue of representing simultaneous causation was discussed in Section 3.8.1. Essentially the idea was to introduce what is called the 'mythical time' to order the events which forces a temporal ordering between them, as if there was a time lapse between them. This approach is feasible as long as we are describing something which *can* somehow be interpreted to be ordered in this way. An example was the application of physical equation in the information theoretical way, such as calculating the force from the mass using Newton's law. However, there are cases where such ordering seem inappropriate. For instance, if we want to represent a causal rule which says, 'when the car is of American configuration, if we press the *exit_button* then the seat will move backwards'. In the CI notation, the causal rule

would look like

$$\Box(t,t,american(X))\wedge\Box(t,t,press(exit\_button,X)\supset\Box(t+1,t+1,move\_seat(backward,X)$$

where *press* and *move_seat* describe such action or movement. It can be argued, however, that the condition $american(X)$ is not time-dependent. It is an attribute of a car which does not (usually) change over time. We might as well treat such conditions as nontemporal conditions, e.g.,

$$american(X)\wedge\Box(t,t,press(exit\_button,X)\supset\Box(t+1,t+1,move\_seat(backward,X)$$

Apart from separating temporal and nontemporal conditions, this has the advantage of not having to keep $american(X)$ in the time-bounded model. We can simply treat its truth condition by looking up a database of nontemporal conditions and kept separate from temporal models. Such an extension would involve rewriting the syntax of causal rules, but may not be too difficult to incorporate.

9.3.2 Computational efficiency

An obvious drawback of this approach to behaviour-oriented design is its computational complexity. Since it is based on abductive model of design, it suffers from the complexity of abductive reasoning which is considered to be NP-hard [Selman & Levesque 90]. The devised abduction framework preferred states that persist backward as long as possible, but if there is no state which has such potential, it is no different from the ordinary abductive mechanism. One way to control it, as we suggested, is by providing some domain dependent heuristics. Although such domain dependent solutions may be available, it is of research interest to investigate other ways, for instance by combining several preference criteria, to reduce complexity of the abduction process.

9.3.3 Integration with multiple specialised reasoning mechanisms

As has been stated, the framework of behaviour-oriented design is not a design method which replaces conventional or other AI approaches to design. Instead, its aim is to provide a facility to represent the dynamic aspect of devices in their specification. It

could comprise a part of an overall design (support) system, or could be implemented as a specialised independent mechanism. An interesting approach is to see how it would be integrated with, for instance, a spatial reasoning based design support tool, providing an extra dimension to it. An issue worth investigating is how to interface these different kinds of information.

Work in functional modelling suggests an integration of various types of modelling methods. For example, a recent paper by Lind[93] suggests that representation of complex process plants in terms of several interrelated levels of abstraction is useful to reduce its complexity in diagnosis and control tasks. Lind's Multilevel Flow Modelling (MFM) employs the strategy of using different representations depending on the focus of the problem (means-end abstraction). While the representations used in MFM's framework seem feasible, it lacks generality in the sense of modularity—it does not offer a framework to integrate specialised reasoning mechanisms from different sources. However, the significance of this family of research is their emphasis on representing a device from multiple perspectives and there is a place in this which causal reasoning should occupy.

Relations to non-AI approaches of design

In this thesis, the studies of design methodologies were focused on those being investigated in the field of AI. While these methodologies have significance in providing a general framework to adapt various techniques of design, there is still a gap between these methodologies and design techniques which are used in practice. For instance, the design of electrical circuits has well established methods developed over the years and the same can be said about various other domains. However, this should not stop us from investigating alternative approaches to design. By studying the design methods from different perspectives, there is a potential to provide techniques which can be applied across multiple domains. Such techniques might prove to be more useful or even powerful. An important thing for those working using AI approaches is while appreciating conventional methods, suggest ways to provide alternative perspectives to problem solving.

To predict the future of a curve is to carry out a certain operation on its past. The true prediction operation cannot be realized by any constructible apparatus; but there are certain operations which bear it a certain resemblance and are, in fact, realizable by apparatus which we can build.

— Norbert Wiener, 1948 [Wiener 65, p.6]

Bibliography

[Aho *et al* 83] A. V. Aho, J. E. Hopfcroft, and J. D. Ullman. *Data structures and algorithms*. Addison-Wesley, London, 1983. ISBN 0-201-00023-7.

[Anderson 38] J. Anderson. The problem of causality. *Australasian Journal of Psycology and Philosophy*, 16:127-142, 1938.

[Baker 89] A. B. Baker. A Simple Solution to the Yale Shooting Problem. In H. J. Levesque R. J. Brachman and R. Reiter, editors, *Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 11-20, May 1989.

[Barrow 84] H. G. Barrow. VERIFY: A Program for Proving Correctness of Digital Hardware Designs. *Artificial Intelligence*, 24:437-491, 1984.

[Bell 91] J. Bell. Extended causal theories. *Artificial Intelligence*, 48:211-224, 1991.

[Berlin 78] I. Berlin. The concept of scientific history. In H. Hardy, editor, *Concepts and categories*, pages 103-142. The Hogarth Press, London, 1978.

[Boddy *et al* 92] M. Boddy, T. Dean, J. Carciofini, and B. Schrag. A semantics for practical temporal reasoning. *Unpublished paper*, 1992. Honeywell Systems & Research Center.

[Chapman 87] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333-377, 1987.

[Charniak & McDermott 85] E. Charniak and D. McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, Massachusetts, 1985.

[Clancey 85] W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27:289-350, 1985.

[Cox & Pietrzykowski 86] P. T. Cox and T. Pietrzykowski. Causes for events: Their computation and applications. In J. Siekmann, editor, *Lecture Notes in Computer Science: Proc. of the 8th International Conference on Automated Deduction*, pages 608-621. Springer-Verlag, 1986.

[Coyne *et al* 90] R. D. Coyne, M. A. Rosenman, A. D. Radford, M. Balachandran, and J. S. Gero. *Knowledge-based Design Systems*. Addison-Wesley, 1990.

[Curd 80] M. V. Curd. The logic of discovery: An analysis of three approaches. In T. Nickels, editor, *Scientific Discovery, Logic and Rationality*, pages 201–219. D. Reidel Publishing Company, Dordrecht and Boston, 1980.

[Davis 84] R. Davis. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence*, 24:347–410, 1984.

[Dean & Boddy 87] T. Dean and M. Boddy. Incremental causal reasoning. In *Proc. AAAI-87*, pages 196–201, 1987.

[Dean & Kanazawa 88] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. In *AAAI-88*, 1988.

[Dean & McDermott 87] T. Dean and D. V. McDermott. Temporal data base management. *Artificial Intelligence*, 32:1–55, 1987.

[deKleer & Brown 86] J. de Kleer and J. S. Brown. Theories of causal ordering. *Artificial Intelligence*, 24:33–62, 1986.

[Dummet 93] M. Dummet. Bringing about the past. In R. Le Poidevin and M. MacBeth, editors, *The Philosopy of Time*, pages 117–133. Oxford University Press, Oxford, England, 1993. ISBN 0-19-823999-8.

[Dym *et al* 92] C. L. Dym, Jr. J. H. Garrett, and D. R. Rehak. Articulating and integrating design knowledge. In D. Finn, editor, *Preliminary Stages of Engineering Analysis and Modelling Workshop, in conjunction with the 2nd International Conference on Artificial Intelligence in Design, Pittsburgh*, pages 2–8, June 1992.

[Etherington & Reiter 83] D. W. Etherington and R. Reiter. On inheritance hierarchies with exceptions. In *Proc. AAAI-83*, pages 104–108, 1983.

[Etherington *et al* 91] D. W. Etherington, S. Kraus, and D. Perlis. Nonmonotonicity and the scope of reasoning. *Artificial Intelligence*, 52:221–261, 1991.

[Feynman 67] R. Feynman. *The Character of Physical law*. MIT Press, Cambridge, Massachusetts, 1967. ISBN 0-262-56003-8.

[Fikes *et al* 71] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:198–208, 1971.

[Fikes *et al* 72] R. E. Fikes and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Finger & Dixon 89] S. Finger and J. R. Dixon. A review of research in mechanical engineering design. part ii: Representations, analysis, and design for the life cycle. *Research in Engineering Design*, 1:121–137, 1989.

[Finger & Genesereth 85] J. J. Finger and M. R. Genesereth. Residue a deductive approach to design synthesis. Technical Report HPP-85-1, Dept of Computer Science, Stanford University, January 1985.

[Finn *et al* 92] D. P. Finn, J. B. Grimson, and N. M. Harty. An intelligence modelling assistant for preliminary analysis in design. In J. S. Gero, editor, *Artificial Intelligence in Design '92*, pages 579–596. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

[Forbus 84] K. D. Forbus. Qualitative Process Theory. *Artificial Intelligence*, 24:85–168, 1984.

[Freeman & Newell 71] P. Freeman and A. Newell. A model for functional reasoning in design. In *Proc. IJCAI-71*, pages 621–640, 1971.

[Friedrich & Lackinger 91] G. Friedrich and F. Lackinger. Diagnosing temporal misbehavior. In *IJCAI-91*, pages 1116–1122, 1991.

[Gero 90] J. Gero. Design prototypes: A knowledge representation schema for design". *AI Magazine*, 11(4):27–36, 1990.

[Gibson 83] J. R. Gibson. *Electronic Logic Circuits (Second Edition)*. Edward Arnold, London, 1983. ISBN 0-7131-3491-7.

[Hanks & McDermott 86] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proc. AAAI-86*, pages 328–333, 1986.

[Hanson 58] N. R. Hanson. *Patterns of Discovery*. Cambridge University Press, England, 1958.

[Hayes 84] P. J. Hayes. Naive physics 1 – ontology for liquids. In *Formal Theories of the Commonsense World*. Ablex, Norwood, New Jersey, 1984.

[Hayes-Roth *et al* 83] B. Hayes-Roth, D. A. Waterman, and D. Lenat. *Building Expert Systems*. Addison-Wesley, Reading, Massachusetts, 1983.

[Hughes & Cresswell 72] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen and Co., London, 1972. ISBN 0-416-29460-X.

[Hume 39] D. Hume. *A Treatise of Human Nature*. L.A. Selby-Bigge and N. H. Nidditch (eds), Oxford University Press, Oxford (1978), 1739.

[Iwasaki & Chandrasekaran 92] Y. Iwasaki and B. Chandrasekaran. Design verification through function- and behavior- oriented representations: bridging the gap between function and behavior. In J. S. Gero, editor, *Artificial Intelligence in Design '92, Pittsburgh*, pages 597–616. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992. ISBN 0-7932-1799-8.

[Iwasaki & Simon 86a] Y. Iwasaki and H. Simon. Causality in Device Behavior. *Artificial Intelligence*, 29:3–32, 1986.

[Iwasaki & Simon 86b] Y. Iwasaki and H. Simon. Theories of causal ordering: Reply to dekleer and brown. *Artificial Intelligence*, 29:63–67, 1986.

[Kant 81] I. Kant. *Critique of Pure Reason*. Macmillan, London, 1881. (original publication 1781).

[Kautz 86] H. Kautz. The logic of persistence. In *Proc. AAAI-86*, 1986.

[Konolige 92] K. Konolige. Using default and causal reasoning in diagnosis. In *Principles of Knowledge Representation and Reasoning, Proc. of the Third International Conference (KR'92)*, pages 509–520, 1992.

[Kuipers 86] B. Kuipers. Qualitative Simulation. *Artificial Intelligence*. 29:289–338, 1986.

[Lawson 90] B. Lawson. *How Designers Think (2nd Ed.)*. Butterworth Architecture. Oxford, England. 1990. ISBN 0-7506-0268-6.

[Levesque 89] H. J. Levesque. A knowledge level account of abduction (preliminary version). In *IJCAI-89*, 1989.

[Lewis 73] D. Lewis. Causation. *Journal of Philosophy*, 70:556–567, 1973.

[Lind 93] M. Lind. Mutilevel flow modeling. In *AAAI93 Workshop on Functional Reasoning*, Washington D.C., Jul 1993.

[Loui 88] R. P. Loui. Computing reference classes. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, pages 273–289. North Holland, Amsterdam, 1988.

[Lukaszewicz 90] W. Lukaszewicz. *Non-monotonic reasoning: formalization of commonsense reasoning*. Ellis Horwood Ltd., Chichester, England, 1990. ISBN 0-13-624446-7.

[MacCallum *et al* 92] K. J. MacCallum, B. Yu, A. Frederiksen, and D. McGregor. A system for supporting design configuration. In J. S. Gero, editor *Artificial Intelligence in Design '92, Pittsburgh*, pages 3–19, Dordrecht, The Netherlands, 1992. Kluwer Academic Publishers. ISBN 0-7932-1799-8.

[Mackie 74] J. L. Mackie. *The Cement of the Universe: a study of causation*. Oxford University Press, Oxford, 1974.

[March 76] L. March. The logic of design and the question of value. In L. March, editor, *The Architecture of Form*, pages 1–40. Cambridge University Press, Cambridge, Massachusetts, 1976.

[McCarthy & Hayes 69] J. M. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[McCarthy 63] J. M. McCarthy. Situations, actions and causal laws. Technical Report Memo 2, Stanford Artificial Intelligence Project, 1963.

[McCarthy 80] J. M. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarthy 86] J. M. McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[McDermott 78] D. McDermott. Planning and Acting. *Congnitive Science*, 2:71–109, 1978.

[McTaggart 27] J. M. E. McTaggart. *The Nature of Existence*. Cambridge University Press, Cambridge, England, 1927.

[Mill 43] J. S. Mill. *A System of Logic*. J. W. Parker, London (2nd Edition, 1846), 1843.

[Moore 85] R. Moore. Semantic considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Morgenstern & Stein 88] L. Morgenstern and L. A. Stein. Why things go wrong: A formal theory of causal reasoning. In *IJCAI-88*, pages 518–523, 1988.

[Nakata 92a] K. Nakata. An application of temporal logic for representation and reasoning about design. In *Applied Logic Conference: Logic At Work, Amsterdam*, Dec 1992.

[Nakata 92b] K. Nakata. Behavioural specification with nonmonotonic temporal logic. In D. Finn, editor, *Preliminary Stages of Engineering Analysis and Modelling Workshop, in conjunction with the 2nd International Conference on Artificial Intelligence in Design, Pittsburgh*, pages 41–45, June 1992.

[Nakata 94a] K. Nakata. An application of temporal logic for representation and reasoning about design. In M. Masuch and L. Polos, editors, *Knowledge Representation and Reasoning Under Uncertainty*, chapter 9, pages 131–145. Springer-Verlag, Lecture Notes in Artificial Intelligence 808, Berlin, 1994.

[Nakata 94b] K. Nakata. Nonmonotonic qualitative reasoning. In F. Anger, editor, *AAAI-94 Workshop on Spatial and Temporal Reasoning*, July 1994.

[Ng & Mooney 92] H. T. Ng and R. J. Mooney. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *Principles of Knowledge Representation and Reasoning, Proc. of the Third International Conference (KR'92)*, pages 499–508, 1992.

[Peirce 23] C. S. Peirce. *Chance, love and logic; philosophical essays*. Kegan Paul, Trench, Trubner & Co., London, England, 1923.

[Peirce 55] C. S. Peirce. Abduction and induction. In J. Buchler, editor, *Philosophical Writings of Peirce*. Dover Publications Inc., New York, 1955.

[Pople 77] H. Pople. The formation of composite hypotheses in diagnostic problem solving. In *Proceddings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1030–1037, 1977.

[Rayner 89] M. Rayner. Did newton solve the "extended prediction problem"? In *Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 381–385, 1989.

[Reggia *et al* 83] J. Reggia, D. Nau, and P. Wang. Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies*, 19:437–460, 1983.

[Reiter & Criscuolo 81] R. Reiter and G. Criscuolo. On interacting defaults. In *Proc. IJCAI-81*, pages 270–276, 1981.

[Reiter 78] R. Reiter. On Closed-World Data Bases. In *Logic and Databases*. Plenum Press, New York, 1978.

[Reiter 80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Richards *et al* 92] B. Richards, I. Kraan, and B. Kuipers. Automatic abduction of qualitative models. Technical Report AI92–171, Department of Computer Sciences, University of Texas at Austin, 1992.

[Russell 63] B. Russell. On the notion of cause. In *Mysticism and logic*, pages 132–151. George Allen & Unwin Ltd, London, 1963.

[Sacerdoti 75] E. D. Sacerdoti. The nonlinear nature of plans. In *IJCAI-75*, pages 206–214, 1975.

[Sandewall 92a] E. Sandewall. Features and fluents: A systematic approach to the representation of knowledge about dynamical systems. Technical Report LiTH-IDA-R-92-30, Department of Computer and Information Science, Linköping University, 1992.

[Sandewall 92b] E. Sandewall. The range of applicability of nonmonotonic logics for the inertia problem, 1992. Unpublished manuscript.

[Selman & Levesque 90] B. Selman and H. J. Levesque. Abductive and default reasoning: A computational core. In *AAAI-90, Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 343–348, 1990.

[Sembugamoorthy & Chandrasekaran 86] V. Sembugamoorthy and B. Chandrasekaran. Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In J. L. Kolodner and C. K. Riesbeck, editors, *Experience, Memory and Reasoning*, pages 47–73. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Shanahan 89] M. Shanahan. Prediction is deduction but explanation is abduction. In *IJCAI-89*, pages 1055–1060, 1989.

[Shoemaker 69] S. Shoemaker. Time without change. *Jouranal of Philosophy*, 66:363–381, 1969.

[Shoham 88] Y. Shoham. *Reasoning about Change*. The MIT Press, Cambridge, Massachusetts, 1988. ISBN 0–262–19269–1.

[Shoham 91] Y. Shoham. Remarks on Simon's comments. *Cognitive Science*, 15:301–303, 1991.

[Shortliffe 76] E.H. Shortliffe. *Computer-Based Medical Consultations: MYCIN.* Elsevier, New York, 1976.

[Simon 69] H. A. Simon. *The Sciences of the Artificial.* The MIT Press, Cambridge, Massachusetts, 1969. ISBN 0-262-19193-8.

[Simon 73] H. A. Simon. The structure of ill-structured problems. *Artificial Intelligence*, 4:181-201, 1973.

[Simon 91] H. A. Simon. Nonmonotonic reasoning and causation: Comment. *Cognitive Science*, 15:293-300, 1991.

[Smith 92] I. Smith. Analysing designs created with preferences and defaults. In D. Finn, editor, *Preliminary Stages of Engineering Analysis and Modelling Workshop, in conjunction with the 2nd International Conference on Artificial Intelligence in Design, Pittsburgh*, pages 59-62, June 1992.

[Smith *et al* 91] I. F. C. Smith, D. Haroud, and B. Faltings. Methods for improving the performance of design systems. In J. Gero, editor, *Artificial Intelligence in Design '91*, pages 467-483. Butterworh-Heinemann, Oxford, 1991.

[Smithers 87] T. Smithers. AI-based design v geometry-based design or why design cannot be supported by geometry alone. *DAI Discussion Paper* No.53, Department of Artificial Intelligence, University of Edinburgh, 1987.

[Stroulia *et al* 92] E. Stroulia, M. Shankar, A. Goel, and L. Penberthy. A model-based approach to blame-assignment in design. In J. S. Gero, editor, *Artificial Intelligence in Design '92, Pittsburgh*, pages 519-537. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992. ISBN 0-7932-1799-8.

[Sturges 92] R. H. Sturges. A computational model for conceptual design based on function logic. In J. S. Gero, editor, *Artificial Intelligence in Design '92, Pittsburgh*, pages 757-772. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992. ISBN 0-7932-1799-8.

[Tate 76] A. Tate. Project planning using a hierarchic nonlinear planner. *DAI Research Report* No.25, Department of Artificial Intelligence, University of Edinburgh, 1976.

[Touretzky 84] D. Touretzky. Implicit ordering of defaults in inheritance systems. In *Proc. AAAI-84*, pages 322-325, 1984.

[Wiener 65] N. Wiener. *Cybernetics: or control and communication in the animal and the machine.* The MIT Press, Cambridge, Massachusetts, second edition, 1965.

[Wilkins 87] D. E. Wilkins. Using causal rules in planning. Technical Report Techinical Note 410R, SRI International, Menlo Park, California, July 1987.

[Williams 84] B. C. Williams. Qualitative analysis of MOS circuits. *Artificial Intelligence*, 24:281-346, 1984.

[Yoshikawa 80] H. Yoshikawa. General design theory and a cad system. In *Proc. of the IFIP WG5.2/5.3 Working Conference, Tokyo*, page 28. North-Holland, Amsterdam, 1980.

Appendix A

Technicalities from Shoham's thesis

Described here are some of the important definitions and theorems extracted from Shoham's thesis. The page numbers indicates those in [Shoham 88].

Chronological minimization

Definition (Chronologically smaller) (p.98) Let $S = p_1, ..., p_n$ be a set of primitive propositions, and M_1 and M_2 two interpretations. M_2 is *chronologically smaller in* S than M_1 (written $M_1 \sqsubset_S M_2$) if there exists a time t_0 such that

1. for all $p \in S$ and all $t_1, t_2 \leq t_0$ (w.r.t. the global interpretation of time), if $M_2 \models \text{TRUE}(t_1, t_2, p)$ then also $M_1 \models \text{TRUE}(t_1, t_2, p)$, and
2. there exists a $p \in S$ and a $t \leq t_0$ such that $M_1 \models \text{TRUE}(t, t_0 p)$ but $M_2 \not\models \text{TRUE}(t, t_0, p)$.

TK: a monotonic logic of temporal knowledge (pp.102–105)

Syntax The syntax of TK is the syntax of the propositional classical interval logic, augmented by the modal operator $\Box$.

Specifically, given p: a set of primitive propositions, TV: a set of variables, TC: the set $\{..., -3, -2, -1, 0, 1, 2, 3, ...\}$ (i.e., the standard representation of the integers), and U: $TC \cup TV$, the set of well-formed formulas of TK is defined inductively as follows:

1. If $u_1 \in U$ and $u_2 \in U$ then $u_1 = u_2$ and $u_1 \preceq u_2$ are wffs.

2. If $u_1 \in U$, $u_2 \in U$, and $p \in P$ then $\text{TRUE}(u_1, t_2, p)$ is a wff.
3. If φ_1 and φ_2 are wffs then so are $\varphi_1 \wedge \varphi_2$, $\neg\varphi_1$, and $\Box\varphi_1$.
4. If φ is a wff and $v \in V$, then $\forall v\varphi$ is also a wff.

We assume the usual definitions of $\vee$,$\supset$,$\equiv$,$\exists$, and so on. As usual, $\Diamond$ is defined by $\Diamond\varphi \equiv \neg\Box\neg\varphi$.

Semantics In TK, a *Kripke interpretation* is a set of infinite "parallel" time lines, all sharing the same interpretation of time: a "synchronized" copy of the integers. Each world describes an entire possible course of the universe, and so over the same time interval, but in different worlds, different facts are true.

More formally we make the following definitions $\mathcal{N}$ is used to denote the integers with $\leq$). A *Kripke interpretation* is a pair $\langle W, M\rangle$ where W is a nonempty universe of *(possible) worlds*, and M is a meaning function $M : P \rightarrow 2^{W\times\mathcal{N}\times\mathcal{N}}$. As in the nonmodal case, we require that $\langle w, t_1, t_2\rangle \in M(p)$ iff $\langle w, t_2, t_1\rangle \in M(p)$, again reflecting the intuition that a pair of time points denotes a single interval.

A *variable assignment* is a function $VA : TV \rightarrow \mathcal{N}$. Also, if $u \in U$ then we define $VAL(u)$ to be $VA(u)$ if $u \in TV$, and the standard interpretation of u if $u \in TC$.

A Kripke interpretation $KI = \langle W, M\rangle$ and a world $w \in W$ satisfy a formula φ under the variable assignment VA (written $KI, w \models \varphi[VA]$) under the following conditions.

- $KI, w \models u_1 = u_2[VA]$ iff $VAL(u_1) = VAL(u_2)$.
- $KI, w \models u_1 \preceq u_2[VA]$ iff $VAL(u_1) \leq VAL(u_2)$.
- $KI, w \models \text{TRUE}(u_1, u_2, p)[VA]$ iff $\langle w, VAL(u_1), VAL(u_2)\rangle \in M(p)$.
- $KI, w \models (\varphi_1 \wedge \varphi_2)[VA]$ iff $KI, w \models \varphi_1[VA]$ and $KI, w \models \varphi_2[VA]$.
- $KI, w \models (\neg\varphi)[VA]$ iff $KI, w \not\models \varphi[VA]$.
- $KI, w \models (\forall v\varphi)[VA]$ iff $KI, w \models \varphi[VA']$ for all VA' that agree with VA everywhere except possibly on v.
- $KI, w \models \Box\varphi[VA]$ iff $KI, w' \models \varphi[VA]$ for all $w' \in W$.

With the exception of the special nature of time, the logic TK is a perfectly standard modal logic. Consequently, the following definitions are also standard. A Kripke interpretation $KI = \langle W, M\rangle$ and a world $w \in W$ are a *model* for a formula φ (written $KI, w \models \varphi$) if $KI, w \models \varphi[VA]$ for any variable assignment VA. A wff is *satisfiable* if it has a model, and *valid* if its negation has no model. φ_1 *entails* φ_2 iff φ_2 is satisfied by all models of φ_1.

CI: a nonmonotonic logic of temporal knowledge (pp.105–106)

Definition (Base wffs) *Base wffs* are those containing no occurrence of the modal operator.

Atomic base sentences are those in the form TRUE(t_1, t_2, p) denoting that the proposition p is true in the temporal interval t_1 to t_2. It is assumed that $t_1 \leq t_2$ by convention.

Definition ("Latest time point") The *latest time point (ltp)* of a base formula is the latest time point mentioned in it (that is, the latest chronologically, not the last syntactically). Formally, it is defined as follows.

1. The *ltp* of TRUE(t_1, t_2, p) is $max\{t_1, t_2\}$.
2. The *ltp* of $\varphi_1 \wedge \varphi_2$ is the latest (w.r.t. the standard interpretation of time) between the *ltp* of φ_1 and the *ltp* of φ_2.
3. The *ltp* of $\neg\varphi$ is the *ltp* of φ.
4. The *ltp* of $\forall v\varphi$ is the earliest among the *ltp*s of all φ' which result from substituting in φ a time-point symbol for all free occurrences of v, or $-\infty$ if there is no such earliest *ltp*.

Definition (Chronologically more ignorant) A Kripke interpretation M_2 is *chronologically more ignorant* than a Kripke interpretation M_1 (written $M_1 \sqsubset_{ci} M_2$) if there exists a time t_0 such that

1. for any base sentence φ whose *ltp* $\leq t_0$, if $M_2 \models \Box\varphi$ then also $M_1 \models \Box\varphi$, and
2. there exists some base sentence φ whose *ltp* is t_0 such that $M_1 \models \Box\varphi$ but $M_2 \not\models \Box\varphi$.

Definition (Chronologically maximally ignorant) M is said to be *chronologically maximally ignorant* (or cmi) model of φ if $M \models_{\sqsubset_{ci}} \varphi$, that is, if $M \models \varphi$ and there is no other M' such that $M' \models \varphi$ and $M \sqsubset_{ci} M'$.

Definition The logic of chronological ignorance CI, is the nonmonotonic logic $TK_{\sqsubset_{ci}}$.

Causal theories

Definition (Time-bounded Kripke interpretation) (p.111) A *time-bounded Kripke interpretation* M/t is a structure which can be viewed as an incomplete Kripke interpretation. Like a Kripke interpretation it assigns a truth value to atomic prepositions, but only to those whose *ltp* $\leq t$. The truth value of an arbitrary sentence whose *ltp* $\leq t$ is also determined in M/t, according to the usual compositional rules. It is easy to

see that this is well defined, since, by the semantics of the logic and by the definition of an *ltp*, the truth value of a sentence whose $ltp \leq t$ does not depend on any sentence whose $ltp \leq t$.

For such a sentence φ whose $ltp \leq t$, we make the obvious definition of what it means for φ to be satisfied by M/t (written $M/t \models \varphi$).

M/t *partially satisfies* a theory Ψ if M/t satisfies all members of Ψ whose $ltp \leq t$.

Theorem (The "unique cmi model" theorem) (pp.112-113) If Ψ is any causal theory, then

1. Ψ has a cmi model, and
2. if M_1 and M_2 are both cmi models of Ψ, and φ any base sentence, then $M_1 \models \Box\varphi$ iff $M_2 \models \Box\varphi$.

   Proof: It is sufficient to construct a model M for Ψ, and show that M is chronologically more ignorant than any model for Ψ which differs from M on the truth value of some sentence $\Box\varphi$, where φ is a base sentence.

   The construction starts with some time-bounded interpretation M/t_0, and augments it to later time points iteratively:

   (a) Let t_0 be time point preceding the *ltp* of any sentence φ such $\Theta \supset \Box\varphi$ is a boundary condition (by assumption such a point exists). For any base tautology φ whose $ltp \leq t_0$, let $M/t0 \models \Box\varphi$. For any other base wff φ whose $ltp \leq t_0$, let $M/t0 \not\models \Box\varphi$.
   Notice that M/t_0 (partially) satisfies the boundary conditions of Ψ vacuously (since the *ltp*s of all boundary conditions are greater than t_0), and (partially) satisfies all the causal rules (since if their *ltp* is no later than t_0, then their antecedents are falsified).

   (b) We now specify how to augment M/t to $M/t+1$ for any t. Let
   $\text{CONSEQUENTS}_{t+1} = \{ \Box(t', t+1, x) : \Phi \wedge \Theta \supset \Box(t', t+1, x) \in \Psi$, and $M/t \models \Phi \wedge \Theta \}$
   $M/t+1$ is obtained by making all wffs in CONSEQUENTS_{t+1} and all their tautological consequents true, and for any other base wff φ' whose *ltp* is $t+1$, making $\Box\varphi'$ false.

   Note that by the assumption about causal theories, CONSEQUENTS_{t+1} does not contain both a $\Box\varphi$ and $\Box\neg\varphi$, and therefore this construction introduces no inconsistency among knowledge of atomic base sentences. But this means that it introduces no inconsistency at all, since the totality of known base sentences whose *ltp* is $\leq t+1$ are the tautological consequences of the atomic ones.

   Clearly, M_∞/∞ (or simply M) is a model for Ψ. To conclude the proof, it remains to show that if a model M' differs from M on the truth value of $\Box\varphi$ for

some base sentence φ, then M' is not a cmi model. To prove that, suppose that some model M' of Ψ indeed differs from M on the truth value of $\Box\varphi$ for some base sentence φ.

1. If $M' \models \Box\varphi$ for some nontautological base sentence φ whose $ltp \leq t_0$, then M' is chronologically less ignorant than M.
2. Otherwise, there is an earliest t such that $t_0 \leq t$, and such that for some base sentence φ whose ltp is $t+1$, M and M' differ on the truth value of $\Box\varphi$. There are two possible cases:
   (a) $M \models \Box\varphi$ and $M' \not\models \Box\varphi$. By the construction procedure, if $M \models \Box\varphi$ then there exists a sentence $\Phi \wedge \Theta \supset \Box\varphi \in \Psi$, such that the ltps of the base sentences in Φ and Θ are $\leq t$, and such that $M \models \Phi \wedge \Theta$. Since M and M' agree on the knowledge of all base sentences whose $ltp \leq t$, it must be the case that also $M' \models \Phi \wedge \Theta$. But since $M' \not\models \Box\varphi$, M' cannot be a model for Ψ; contradiction.
   (b) $M' \models \Box\varphi$ and $M \not\models \Box\varphi$. From the first case it is known that for any φ' whose ltp is $t+1$, if $M \models \Box\varphi'$ then also $M' \models \Box\varphi'$. But these two facts together imply that M' is chronologically less ignorant than M.
   □

Inertial theories

Theorem (The "unique"-model property) (pp.136-140) If $\Psi = \Psi_1 \cup \Psi_2 \cup \Psi_3$ is any inertial theory as above, then

1. Ψ has a cmi model, and
2. if M_1 and M_2 are both cmi models of Ψ, and φ any base sentence, then $M_1 \models \Box\varphi$ iff $M_2 \models \Box\varphi$.

Proof: The flavour of the proof is identical to that of the proof of the analogous theorem for causal theories. We will construct a model M for Ψ, and show that M is chronologically more ignorant than any model for Ψ which differs form M of the truth values of some sentence $\Box\varphi$, where φ is a base sentence. Again, the construction starts with some time-bounded interpretation M/t_0, and augments it to later time points iteratively. The difference between the two proofs will be in the inductive construction step. Intuitively, at each point we will determine not only what effects are actually still manifested at that time. We will therefore compute not only the CONSEQUENTS$_t$ set at each time point, the "effects" which terminate at t, but also the POTENTIALS$_t$ set, which identify the potential histories which are still active at time t. More precisely, $\langle t_1, t_2, p\rangle$ will be in POTENTIALS$_t$ just in case there is a potential history POTEN(t_1, t_2, p) such that $t_1 \leq t \leq t_2$, and that potential history in

fact does not terminate before t. Of the potential histories in POTENTIALS$_t$, some will actually terminate at t. This might be because t is the upper bound on the duration of the potential (the NATURAL-DEATH$_t$ set), or because extending it to $t+q$ causes a contradiction (the CLIPPED$_t$ set).

Since the identity of the potential histories ending at time t depends in part on known propositions whose *ltp* is $t+1$, determining them will "lag" one time instant behind determining the other known propositions. In other words, strictly speaking we will not simply extend M/t to $M/t+1$ in each iteration. Rather, at that stage we will determine the known temporal propositions whose *ltp* is no later than $t+1$ *except for the propositions denoting potential histories, or the* p- *propositions.* In addition, at this stage we will complete the M/t model constructed in the previous iteration by determining the known p- propositions that ended at time t.

$M/t+1$ is obtained by making all wffs in CONSEQUENTS$_{t+1}$ and all their tautological consequents true, and for any other base wff φ whose *ltp* is $t+1$ (other than p- wffs), making $\Box\varphi$ false. (The details of the construction of the sets is omitted.)

$M/t+1$ determines which base sentences are known whose *ltp* is $\leq t+1$ — except for containing p- propositions. However, at his time one can determine all the known p- sentences with *ltp* t. The known atomic p- sentences whose *ltp* is t are exactly those in NATURAL-DEATH$_t$ $\cup$ CLIPPED$_t$, which, together with closure of knowledge under tautological consequence, completely determines *all* the base sentences which are known in M/t.

As in the case of causal theories, here too it can be seen that the construction of $M/t+1$ introduces no inconsistency.

Since the construction of $M/t+1$ never violates any sentence in Ψ, it follows that M/∞ (or simply M) is a model for Ψ. To conclude the proof, it remains to show that if a model m' differs from M on the truth value of $\Box\varphi$ for some base sentence φ, then M' is chronologically less ignorant than M. Since this part of the proof is even more tedious and unilluminating than the first part of the proof, it is eliminated here. For details, see [Shoham 88, Appendix B].

Appendix B

Causal theories for selected sample problems

To treat causal theories using the programs attached in Appendix C, the logical expressions of causal rules needs to be transcribed using corresponding user defined Prolog operators as follows:

symbol	Prolog operator
$\Box$	`nec`
$\Diamond$	`pos`
$\supset$	`cause`
$\wedge$	`and`
$\neg$	`not`
POTEN	`'POTEN'`
PROJECT	`'PROJECT'`
B-PROJECT	`'B-PROJECT'`
∞	`inf`
`p-`	`p-`

- Temporal variables such as $t, t+1, t_0, t_1$ may be written in the same manner as `t, t+1, t0, t1`.
- The characters `t`, `t+`n, `t`n (n: integer) are reserved for temporal variables.

For example, a causal theory

$\Box(1, 1, p)$.
$\Box(t, t, q) \wedge \Diamond(t, t, \neg r) \supset \Box(t+1, t+1, s)$.
$\Box(t, t, u) \supset \text{POTEN}(t, \infty, p-v)$.
$\text{PROJECT}(t_1, p-v, t, t, v)$.

is transcribed as

```
nec(1,1,p).
nec(t,t,q) and pos(t,t,not r) cause nec(t+1,t+1,s).
nec(t,t,u) cause 'POTEN'(t,inf,p-v).
'PROJECT'(t1,p-v,t,t,v).
```

B.1 Simple oscillator problem

The causal theory for the 'Simple oscillator problem' (Chapter 3) is formulated as follows.

Causal theory

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                  %%
%%    Simple Oscillator Problem     %%
%%                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Boundary conditions *************************************

% 1   The initial position is "stretched"
nec(1,1,stretched).

% 2   The initial velocity is 0
nec(1,1,v=0).

% 3   The initial acceleration is 0
nec(1,1,a=0).

% 4   The initial force is 0
nec(1,1,f=0).

% 5   The block is initially held
nec(1,1,held).

% 6   The block is released at time-point 3
nec(3,3,released).

% 6A  The spring breaks at time-point 5
%nec(5,5,springbreak).

%% Persistence conditions *********************************

% 7   The block is held unless released
nec(t,t,held) and
   pos(t,t,not released) cause
   nec(t+1,t+1,held).

% 8   The force is 0 while it is held
nec(t,t,held) and
   pos(t,t, not released) and
   pos(t,t, not springbreak) cause
   nec(t+1,t+1,f=0).

% 9   The position does not change unless the block has
%     non-zero velocity  (axioms 9 to 13)
nec(t,t,stretched) and
   pos(t,t, not (v=-)) cause
   nec(t+1,t+1,stretched).
```

```
%10
nec(t,t,p=+) and
   pos(t,t, not (v=-)) and
   pos(t,t, not (v=+)) cause
   nec(t+1,t+1,p=+).

%11
nec(t,t,p=0) and
   pos(t,t, not (v=-)) and
   pos(t,t, not (v=+)) cause
   nec(t+1,t+1,p=0).

%12
nec(t,t,p=-) and
   pos(t,t, not (v=-)) and
   pos(t,t, not (v=+)) cause
   nec(t+1,t+1,p=-).

%13
nec(t,t,compressed) and
   pos(t,t, not (v=+)) cause
   nec(t+1,t+1,compressed).

%14   The velocity does not change unless the acceleration is
%     non-zero  (axioms 14 to 16)
nec(t,t,v=0) and
   pos(t,t, not (a=-)) and
   pos(t,t, not (a=+)) cause
   nec(t+1,t+1,v=0).

%15
nec(t,t,v=-) and
   pos(t,t, not (a=-)) and
   pos(t,t, not (a=+)) cause
   nec(t+1,t+1,v=-).

%16
nec(t,t,v=+) and
   pos(t,t, not (a=-)) and
   pos(t,t, not (a=+)) cause
   nec(t+1,t+1,v=+).

%17   The acceleration does not change unless the force is
%     non-zero  (axioms 17 to 19)
nec(t,t,a=0) and
   pos(t,t, not (f=-)) and
   pos(t,t, not (f=+)) cause
   nec(t+1,t+1,a=0).

%18
nec(t,t,a=-) and
   pos(t,t, not (f=0)) and
   pos(t,t, not (f=+)) cause
   nec(t+1,t+1,a=-).

%19
nec(t,t,a=+) and
   pos(t,t, not (f=-)) and
   pos(t,t, not (f=0)) cause
   nec(t+1,t+1,a=+).

%% Causal rules ******************************************

%20   The block will initially have a negative force at the
%     release
nec(t,t,f=0) and
   nec(t,t,release) and
   pos(t,t,not springbreak) cause
   nec(t+1,t+1,f=-).

%21   If the spring breaks, the force will be zero
nec(t,t,springbreak) cause
   nec(t+1,t+1,f=0).
```

```
%22   Once the spring breaks, it remains broken unless it is
%     fixed
nec(t,t,springbreak) and
   pos(t,t,not fixed) cause
   nec(t+1,t+1,springbreak).

%23   Hooke's law:  f=kx    (axioms 23 to 27)
nec(t,t,stretched) and
   nec(t,t,released) and
   pos(t,t, not springbreak) cause
   nec(t+1,t+1,f=-).

%24
nec(t,t,p=+) and
   pos(t,t, not held) and
   pos(t,t, not springbreak) cause
   nec(t+1,t+1,f=-).

%25
nec(t,t,p=0) and
   pos(t,t, not held) and
   pos(t,t, not springbreak) cause
   nec(t+1,t+1,f=0).

%26
nec(t,t,p=-) and
   pos(t,t, not held) and
   pos(t,t, not springbreak) cause
   nec(t+1,t+1,f=+).

%27
nec(t,t,compressed) and
   pos(t,t, not held) and
   pos(t,t, not springbreak) cause
   nec(t+1,t+1,f=+).

%28   Newton's law:  f=ma    (axioms 28 to 30)
nec(t,t,f=+) cause
   nec(t+1,t+1,a=+).

%29
nec(t,t,f=0) cause
   nec(t+1,t+1,a=0).

%30
nec(t,t,f=-) cause
   nec(t+1,t+1,a=-).

%31   Transposition w.r.t. the velocity   (axioms 31 to 38)
nec(t,t,stretched) and
   nec(t,t,v=-) cause
   nec(t+1,t+1,p=+).

%32
nec(t,t,p=+) and
   nec(t,t,v=-) cause
   nec(t+1,t+1,p=0).

%33
nec(t,t,p=+) and
   nec(t,t,v=+) cause
   nec(t+1,t+1,stretched).

%34
nec(t,t,p=0) and
   nec(t,t,v=-) cause
   nec(t+1,t+1,p=-).

%35
nec(t,t,p=0) and
   nec(t,t,v=+) cause
   nec(t+1,t+1,p=+).
```

```
%36
nec(t,t,p=-) and
   nec(t,t,v=-) cause
   nec(t+1,t+1,compressed).

%37
nec(t,t,p=-) and
   nec(t,t,v=+) cause
   nec(t+1,t+1,p=0).

%38
nec(t,t,compressed) and
   nec(t,t,v=+) cause
   nec(t+1,t+1,p=-).

%39   From the definition of velocity in terms of
%     acceleration   (axioms 39 to 44)
nec(t,t,v=-) and
   nec(t,t,a=-) cause
   nec(t+1,t+1,v=-).

%40
nec(t,t,v=-) and
   nec(t,t,a=+) cause
   nec(t+1,t+1,v=0).

%41
nec(t,t,v=0) and
   nec(t,t,a=-) cause
   nec(t+1,t+1,v=-).

%42
nec(t,t,v=0) and
   nec(t,t,a=+) cause
   nec(t+1,t+1,v=+).

%43
nec(t,t,v=+) and
   nec(t,t,a=-) cause
   nec(t+1,t+1,v=0).

%44
nec(t,t,v=+) and
   nec(t,t,a=+) cause
   nec(t+1,t+1,v=+).
```

Prediction

Prediction of the behaviour of the oscillator is obtained by computing the cmi model.

•Case1: spring does not break

```
| ?- prove(osc,18).

...checking the sentences in the theory... OK

  t=  1 :    held
             stretched
             a=0
             f=0
             v=0

  t=  2 :    held
             stretched
             a=0
             v=0

  t=  3 :    held
             released
```

```
            stretched
            a=0
            v=0

  t= 4 :    stretched
            f=-
            a=0
            v=0

  t= 5 :    stretched
            a=-
            v=0

  t= 6 :    stretched
            a=-
            v=-

  t= 7 :    a=-
            v=-
            p=+

  t= 8 :    a=-
            f=-
            v=-
            p=0

  t= 9 :    a=-
            p=-
            v=-
            f=0

  t= 10 :   compressed
            v=-
            a=0
            f=+

  t= 11 :   compressed
            v=-
            a=+
            f=+

  t= 12 :   compressed
            v=0
            a=+
            f=+

  t= 13 :   compressed
            a=+
            f=+
            v=+

  t= 14 :   p=-
            a=+
            f=+
            v=+

  t= 15 :   p=0
            a=+
            f=+
            v=+

  t= 16 :   f=0
            a=+
            p=+
            v=+

  t= 17 :   stretched
            f=-
            a=0
            v=+

  t= 18 :   stretched
            a=-
            v=+

yes
```

```
| ?-
```

•Case2: spring breaks at time 5

```
| ?- prove(osc,18).

...checking the sentences in the theory... OK

  t= 1 :    held
            stretched
            a=0
            f=0
            v=0

  t= 2 :    held
            stretched
            a=0
            v=0

  t= 3 :    held
            released
            stretched
            a=0
            v=0

  t= 4 :    stretched
            f=-
            a=0
            v=0

  t= 5 :    springbreak
            stretched
            a=-
            v=0

  t= 6 :    springbreak
            stretched
            a=-
            v=-
            f=0

  t= 7 :    springbreak
            v=-
            a=0
            f=0
            p=+

  t= 8 :    springbreak
            v=-
            a=0
            f=0
            p=0

  t= 9 :    springbreak
            p=-
            v=-
            a=0
            f=0

  t= 10 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 11 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 12 :   compressed
            springbreak
            v=-
```

```
            a=0
            f=0

  t= 13 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 14 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 15 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 16 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 17 :   compressed
            springbreak
            v=-
            a=0
            f=0

  t= 18 :   compressed
            springbreak
            v=-
            a=0
            f=0

yes
| ?-
```

B.2 Ticketed car scenario

'Ticketed car scenario' (Chapter 4) involves bidirectional sweeping to identify the possible occurrence of the 'ticketing' event.

Causal theory

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                              %%
%%  Ticketed Car Scenario       %%
%%                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1
nec(t,t,park(car)) cause 'POTEN'(t+1,inf,p-parked(car)).

%2
nec(t,t,ticket(car)) cause 'POTEN'(t+1,inf,p-ticketed(car)).

%3
nec(t,t,parked(car)) and
   nec(t,t,ticket(car)) and
   nec(t,t,not ticketed(car)) and
```

```
   nec(t,t,time(day)) cause
   nec(t+1,t+1,ticketed(car)).

%4
nec(1,1,park(car)).

%5
nec(1,3,time(night)).

%6
nec(4,6,time(day)).

%7
nec(7,9,time(night)).

%8
nec(10,12,time(day)).

%9
nec(13,15,time(night)).

%10
'POTEN'(1,inf,p-not ticketed(car)).

%10A
nec(14,14,parked(car)).

%11
nec(14,14,ticketed(car)).

'PROJECT'(t1,p-P,t1,t,P).
```

Forward sweeping

The cmi model is constructed for the causal theory.

```
| ?- prove(tcs,15).

...checking the axioms and the sentences in the theory... OK

**** CHRONOLOGICALLY MINIMUM SET FOR tcs ****

  t= 1 :    park(car)
            time(night)

  t= 2 :    parked(car)
            time(night)

  t= 3 :    parked(car)
            time(night)

  t= 4 :    parked(car)
            time(day)

  t= 5 :    parked(car)
            time(day)

  t= 6 :    parked(car)
            time(day)

  t= 7 :    parked(car)
            time(night)

  t= 8 :    parked(car)
            time(night)

  t= 9 :    parked(car)
            time(night)

  t= 10 :   parked(car)
```

```
            time(day)

  t= 11 :   parked(car)
            time(day)

  t= 12 :   parked(car)
            time(day)

  t= 13 :   parked(car)
            time(night)

  t= 14 :   parked(car)
            ticketed(car)
            time(night)
```

Backward sweeping

A bci set is constructed with backward persistence.

```
| ?- b_prove(tcs).

...checking the axioms and the sentences in the theory... OK

**** BCI SET FOR tcs ****

  t=  1 :   park(car)
            parked(car)
            time(night)

  t=  2 :   parked(car)
            time(night)

  t=  3 :   parked(car)
            time(night)

  t=  4 :   parked(car)
            time(day)

  t=  5 :   parked(car)
            time(day)

  t=  6 :   parked(car)
            time(day)

  t=  7 :   parked(car)
            time(night)

  t=  8 :   parked(car)
            time(night)

  t=  9 :   parked(car)
            time(night)

  t= 10 :   parked(car)
            time(day)

  t= 11 :   parked(car)
            time(day)

  t= 12 :   parked(car)
            ticket(car)
            time(day)

  t= 13 :   parked(car)
            ticketed(car)
            time(night)

  t= 14 :   parked(car)
            ticketed(car)
```

```
time(night)
```

Identifying missing events

From the result of bidirectional sweeping, possible candidate time points for the 'ticketing' event to take place are suggested.

```
| ?- find_event.

  **********   Discrepancies   ************
Time point       forward                    backward
-----------------------------------------------------
    2          not ticketed(car)          ticketed(car)
    3          not ticketed(car)          ticketed(car)
    4          not ticketed(car)          ticketed(car)
    5          not ticketed(car)          ticketed(car)
    6          not ticketed(car)          ticketed(car)
    7          not ticketed(car)          ticketed(car)
    8          not ticketed(car)          ticketed(car)
    9          not ticketed(car)          ticketed(car)
    10         not ticketed(car)          ticketed(car)
    11         not ticketed(car)          ticketed(car)
    12         not ticketed(car)          ticketed(car)
    13         not ticketed(car)          ticketed(car)
    14         not ticketed(car)          ticketed(car)
  *********   end of description   *********

  *** The following event(s) should have occurred ***
At time point 4  ticket(car)
At time point 5  ticket(car)
At time point 6  ticket(car)
At time point 10  ticket(car)
At time point 11  ticket(car)
At time point 12  ticket(car)
```

This suggests that the event *ticket(car)* possibly took place during time points 4 to 6, or 10 to 12.

B.3 SR flip-flop circuit

In SR flip-flop circuit problem (Chapter 8), the causal theory consists of the causal rules for NAND gates and connections between the terminals.

Causal theory

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                           %%
%%  SR Flip-Flop Circuit     %%
%%                           %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% NAND Gate
%1
nec(t,t,input(X,a,1)) and
   nec(t,t,input(X,b,1)) cause
   'POTEN'(t+1,inf,p-output(X,0)).
%2
nec(t,t,input(X,a,0)) and
```

```
   pos(t,t,input(X,b,N)) cause
   'POTEN'(t+1,inf,p-output(X,1)).
%3
nec(t,t,input(X,b,0)) and
   pos(t,t,input(X,a,N)) cause
   'POTEN'(t+1,inf,p-output(X,1)).

%% Prototype configuration
%4
nec(t,t,s(N)) cause
   nec(t+1,t+1,input(nand1,a,N)).
%5
nec(t,t,r(N)) cause
   nec(t+1,t+1,input(nand2,b,N)).
%6
nec(t,t,output(nand1,N)) cause
   'POTEN'(t+1,inf,p-q1(N)).
%7
nec(t,t,output(nand2,N)) cause
   'POTEN'(t+1,inf,p-q2(N)).

'PROJECT'(t1,p-s(X),t,t,s(X)).
'PROJECT'(t1,p-r(X),t,t,r(X)).
'PROJECT'(t1,p-q1(X),t,t,q1(X)).
'PROJECT'(t1,p-q2(X),t,t,q2(X)).
'PROJECT'(t1,p-output(X,N),t,t,output(X,N)).
```

Scenario

The following is the scenario for the 'set' operation, where input to $\overline{R}$ is kept to 1 while $\overline{S}$ is changed from 1 to 0 then back to 1.

```
%% Boundary conditions for Scenario 1

nec(1,3,s(1)).
nec(4,6,s(0)).
nec(7,9,s(1)).
nec(1,9,r(1)).
```

Behavioural specification

The behavioural specification for the 'set' operation is described as follows, which sets the value of Q to 1 and $\overline{Q}$ to 0 after the operation.

```
%% Behavioural specification for Scenario 1

(1,3,s(1)).
(4,6,s(0)).
(7,9,s(1)).
(1,9,r(1)).
(10,12,q1(1)).
(10,12,q2(0)).
```

Sample session

With the above causal theory (prototype configuration), scenario and behavioural specification, a sample design session looks as follows:

```
| ?- go.

>> Enter causal theory file name: srff.

>> Enter specification file name: srff_spec.

>> Enter duration of prediction: 12.

Constructing c.m.i. model...

...checking the axioms and the sentences in the theory... OK

**** CHRONOLOGICALLY MINIMUM SET FOR srff ****

  t= 1 :    r(1)
            s(1)

  t= 2 :    r(1)
            s(1)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 3 :    r(1)
            s(1)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 4 :    r(1)
            s(0)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 5 :    r(1)
            s(0)
            input(nand1,a,0)
            input(nand2,b,1)

  t= 6 :    r(1)
            s(0)
            output(nand1,1)
            input(nand1,a,0)
            input(nand2,b,1)

  t= 7 :    q1(1)
            r(1)
            s(1)
            output(nand1,1)
            input(nand1,a,0)
            input(nand2,b,1)

  t= 8 :    q1(1)
            r(1)
            s(1)
            output(nand1,1)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 9 :    q1(1)
            r(1)
            s(1)
            output(nand1,1)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 10 :   q1(1)
            output(nand1,1)
            input(nand1,a,1)


  t= 11 :   q1(1)
            output(nand1,1)

  t= 12 :   q1(1)
            output(nand1,1)
```

```
Discrepancy occurring at time point 10
    q2(0) should be obtained at this point.

Suggesting new rule to add and prove...

nec(t,t,q1(1))cause nec(t+1,t+1,q2(0))

Rule--->
>> Another candidate?[y/n/p/q]: y.

nec(t,t,output(nand1,1))cause nec(t+1,t+1,q2(0))

Rule--->
>> Another candidate?[y/n/p/q]: n.

Assert new rule:
   nec(t,t,output(nand1,1))cause nec(t+1,t+1,input(nand2,a,1))

Proof--->
>> Another candidate?[y/n/p/q]: n.
```

The first suggestion, `nec(t,t,q1(1)) cause nec(t+1,t+1,q2(0))` was discarded by the designer since it does not seem to be a reasonable causal relation. Once the next suggestion `nec(t,t,output(nand1,1)) cause nec(t+1,t+1,q2(0))` is selected, the system constructs the rules that provide this effect, thus proposing the assertion of the rule `nec(t,t,output(nand1,1)) cause nec(t+1,t+1,input(nand2,a,1))`. We now assert this rule, which suggests the connection between the output terminal of NAND1 and an input terminal a of NAND2. Then rerun the process, which demonstrates the following:

```
| ?- go.

>> Enter causal theory file name: srff.

>> Enter specification file name: srff_spec.

>> Enter duration of prediction: 12.

Constructing c.m.i. model...

...checking the axioms and the sentences in the theory... OK

**** CHRONOLOGICALLY MINIMUM SET FOR srff ****

  t= 1 :    r(1)
            s(1)

  t= 2 :    r(1)
            s(1)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 3 :    r(1)
            s(1)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 4 :    r(1)
            s(0)
            input(nand1,a,1)
            input(nand2,b,1)

  t= 5 :    r(1)
            s(0)
            input(nand1,a,0)
            input(nand2,b,1)

  t= 6 :    r(1)
```

```
            s(0)
            output(nand1,1)
            input(nand1,a,0)
            input(nand2,b,1)

 t= 7 :     r(1)
            s(1)
            output(nand1,1)
            input(nand1,a,0)
            input(nand2,a,1)
            input(nand2,b,1)

 t= 8 :     q1(1)
            r(1)
            s(1)
            output(nand1,1)
            output(nand2,0)
            input(nand1,a,1)
            input(nand2,a,1)
            input(nand2,b,1)

 t= 9 :     q1(1)
            q2(0)
            r(1)
            s(1)
            output(nand1,1)
            output(nand2,0)
            input(nand1,a,1)
            input(nand2,a,1)
            input(nand2,b,1)

 t= 10 :    q1(1)
            q2(0)
            output(nand1,1)
            output(nand2,0)
            input(nand1,a,1)
            input(nand2,a,1)
            input(nand2,b,1)

 t= 11 :    q1(1)
            q2(0)
            output(nand1,1)
            output(nand2,0)
            input(nand2,a,1)

 t= 12 :    q1(1)
            q2(0)
            output(nand1,1)
            output(nand2,0)
            input(nand2,a,1)

No discrepancy.

yes
| ?-
```

This indicates that the first scenario has been fulfilled, and the similar process continues for all the scenarios provided.

Appendix C

Program listings

Programs listed here are the essential parts of the code for each module (i.e., these are *not* complete codes). These are all written in SICStus Prolog syntax.

C.1 Program for computing cmi models

This is an implementation of the cmi model construction described in Chapter 3. In this implementation dealing with inertial rules, it involves the construction of the sets, NEW-POTENTIALS, PARTIAL-CONSEQUENTS, NATURAL-DEATH, CLIPPED, POTENTIALS, CONSEQUENTS, where the set CONSEQUENTS contains the tautological consequents of the cmi model.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%          Implementation of Shoham's logic CI                       %%
%%           --- with first-order features                            %%
%%                                                                    %%
%%          Coded by K. Nakata                                        %%
%%                                                                    %%
%%          July 1991                                                 %%
%%          October 1991 (ver.2)                                      %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%
%% DECLARATIONS %%
%%%%%%%%%%%%%%%%%%%%

:- dynamic 'NEW-POTENTIALS'/2,
           'PARTIAL-CONSEQUENTS'/2,
           'NATURAL-DEATH'/2,
           'CLIPPED'/2,
           'POTENTIALS'/2,
           'CONSEQUENTS'/2,
           'TRUE'/3,
           'FALSE'/3,
           'PROJECT'/5,
           'POTEN'/3,
           cause/2,
           nec/3.

%%%%%%%%%%%%%%%%%
%% OPERATORS %%
%%%%%%%%%%%%%%%%%
```

```
:- op(500,fy,include).          % for file inclusion
:- op(300,fy,not).              % for negations
:- op(400,xfy,and).             % for conjunctions
:- op(500,xfx,cause).           % for "implications"
:- op(600,xfx,&<).              % for the comparison of time points

%%%%%%%%%%%%%%%%%%%%%
%% LIBRARY FILES  %%
%%%%%%%%%%%%%%%%%%%%%

:- [utilities].                 % utility predicates
:- [wff].                       % provides wff checking predicate wft/1

%%%%%%%%%%%%%%%%%%%%%%%%
%% DEFINITION OF &<  %%
%%%%%%%%%%%%%%%%%%%%%%%%

A &< B :-
   (var(A); \+(\+nonts(A))),
   (var(B); \+(\+nonts(B))),!.
A &< B :-
   \+(\+ nonts(A)),
   B == t,!.
A &< B :-
   A == t,
   \+(\+ nonts(B)),!.
_ &< T :-
   T == inf,!.

T &< T :- !.
T &< T+_ :- !.
T-_ &< T :- !.
T-_ &< T+_ :- !.

T+M &< T+N :-
   !,M =< N.
T-M &< T-N :-
   !, M >= N.

A &< B :-
   (
    integer(A);
    (A = MA+NA, integer(MA), integer(NA));
    (A = MA-NA, integer(MA), integer(NA))
   ),
   (
    integer(B);
    (B = MB+NB, integer(MB), integer(NB));
    (B = MB-NB, integer(MB), integer(NB))
   ),
   AA is A,
   BB is B,!,
   AA =< BB.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TOP-LEVEL PREDICATES  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Top-level  ---
%
% prove(+Theory,+Target,+InitialTime,+TerminalTime)
% prove(+Theory,+TerminalTime)
%     starts the process of reasoning by inertial theory for the
%     Theory from InitialTime (default=0) to TerminalTime and
%     prints out the tautological consequents of the events specified
%     as Target (a list, default=all)

prove(Theory,TerminalTime):-
   prove(Theory,all,0,TerminalTime).

prove(Theory,Target,InitialTime,TerminalTime):-
   make_theory(Theory),
   assertz('POTENTIALS'(InitialTime,[])),
   update_sets(InitialTime,TerminalTime),
   StartTime is InitialTime+1,
   write('**** CHRONOLOGICALLY MINIMUM SET FOR '),
```

```
    write(Theory),
    write(' ****'),nl,
    pretty_print(StartTime,TerminalTime,Target),
    clcl.

% make_theory(+Theory)
%     for the Theory (filename) specified, collect each formula in
%     the appropriate lists, and asserting them as Prolog clauses

make_theory(Theory):-
    open(Theory,read,Stream),
    read_theory([],List,Stream),
    uft(List),
    make_phi_clauses(List).

% read_theory(+Sofar,+List,+Stream)
%     read in each clause and collect them into a List

read_theory(Sofar,List,Stream):-
    read(Stream,Formula),
    ((Formula == end_of_file,!,List=Sofar,close(Stream));
     (Formula = include File,!,
      open(File,read,S2),
      read_theory([],List2,S2),
      append(List2,Sofar,NewSofar),
      read_theory(NewSofar,List,Stream));
     (append([Formula],Sofar,NewSofar),read_theory(NewSofar,List,Stream))).

% make_phi_clauses(+List)
%     assert the elements in List in Prolog clausal form

make_phi_clauses([]).
make_phi_clauses([H|T]):-
    H =.. ['PROJECT'|_],!,
    subst(H,R,_),
    R =.. ['PROJECT',TT1,_,TT2,TT3,_],
    assertz((R :- TT1 &< TT2, TT2 &< TT3)),
    make_phi_clauses(T).
make_phi_clauses([H|T]):-
    H =.. [_,_,_,_],!,
    subst(H,R,_),
    R =.. [_,TT1,TT2,_],
    assertz((R :- TT1 &< TT2)),
    make_phi_clauses(T).
make_phi_clauses([H|T]):-
    H =.. [cause,L1,L2],!,
    substand(L1,LL1,TT),
    substand(L2,LL2,TT),
    HH =.. [cause,LL1,LL2],
    LL2 =.. [_,TT1,TT2,_],
    assertz((HH :- TT1 &< TT2)),
    make_phi_clauses(T).

% subst(+Clause,-Result,+Variable)
%     substitutes general time point identification (t, t+1, t+2,...)
%     with the Variable specified, and substitutes the anonymous time
%     points (t1, t2, t3,...) with a variable; the rest is retained
%     e.g. foo(1,t,t+1,t1,bar) --> foo(1,T,T+1,_,bar)

subst(A,R,T):-
    A =.. L,
    substl(L,Rl,T),
    R =.. Rl.

substl([],[],_).
substl([A|R],[A|L],T):-
    var(A),!,
    substl(R,L,T).
substl([t+M|R],[T+M|L],T):-
    !,
    substl(R,L,T).
substl([t-M|R],[T-M|L],T):-
    !,
    substl(R,L,T).
substl([t|R],[T|L],T):-
    !,
    substl(R,L,T).
```

```
subst1([A|R],[_|L],T):-
   nonts(A),!,
   subst1(R,L,T).
subst1([A|R],[A|L],T):-
   subst1(R,L,T).

% substand(+Terms,-Result,+Variable)
%    does the same as subst/3, but for the clauses connected with
%    'and' operators

substand(A and B, AA and BB, TT):-
   !,
   subst(A,AA,TT),
   substand(B,BB,TT).
substand(A,AA,TT):-
   subst(A,AA,TT).

% nonts(?Symbol)
%    succeeds if Symbol is one of the reserved symbols for arbitrary
%    time points, i.e. t0,t1,...,tn (n:integer)

nonts(X):-
   \+ atomic(X),!,
   fail.
nonts(X):-
   name(X,[A|B]),
   name(t,[A]),
   name(Y,B),
   integer(Y).

% update_sets(+CurrentTime,+TerminalTime)
%    updates the sets NEW-POTENTIALS, PARTIAL_CONSEQUENTS, NATURAL-DEATH,
%    CLIPPED, POTENTIALS, CONSEQUENTS in order and recurses the process
%    by incrementing the time step by one until Current_time is equalised
%    with TerminalTime

update_sets(T,T):-
   !,nl,nl,nl.

update_sets(IT,TT):-
   NewIT is IT+1,
   update('NEW-POTENTIALS',NewIT),
   'NEW-POTENTIALS'(NewIT,_),
   update('PARTIAL-CONSEQUENTS',NewIT),
   'PARTIAL-CONSEQUENTS'(NewIT,_),
   update('NATURAL-DEATH',IT),
   'NATURAL-DEATH'(IT,_),
   update('CLIPPED',IT),
   'CLIPPED'(IT,_),
   update('POTENTIALS',NewIT),
   'POTENTIALS'(NewIT,_),
   update('CONSEQUENTS',NewIT),
   'CONSEQUENTS'(NewIT,_),
   makemodel(NewIT),
   update_sets(NewIT,TT).

% pretty_print(+InitialTime,+TerminalTime,+Target)
%    prints out the tautologies for the predicates specified in the list
%    Target; Target = all for all events

pretty_print(T,T,L):-
   !,nl,
   (
    (T >9, write('  t= '));
    (T=<9, write('  t=  '))
   ),
   write(T),write(' :    '),
   print_truth(T,L),
   nl,nl.
pretty_print(IT,TT,all):-
   !,nl,
   (
    (IT >9, write('  t= '));
    (IT=<9, write('  t=  '))
   ),
   write(IT),write(' :    '),
   print_truth(IT,all),
```

```
   ITT is IT+1,
   pretty_print(ITT,TT,all).
pretty_print(IT,TT,L):-
   !,nl,
   (
    (IT >9, write('  t= '));
    (IT=<9, write('  t=  '))
   ),
   write(IT),write(' :    '),
   print_truth(IT,L),
   ITT is IT+1,
   pretty_print(ITT,TT,L).

print_truth(T,all):-
   setof(P,T1^'TRUE'(T1,T,P),S),!,
   form_print(S).
print_truth(T,L):-
   \+ L == all,
   setof(P,T1^(L^('TRUE'(T1,T,P),member(P,L))),S),!,
   form_print(S).
print_truth(_,_):-
   write('---'),nl.

form_print([]):- !.
form_print([H|T]):-
   write(H),nl,tab(13),
   form_print(T).

%%%%%%%%%%%%%%%%%%%%%%
%%  UPDATING SETS  %%
%%%%%%%%%%%%%%%%%%%%%%

% update(+Set,+Time)
%    updates the Set at the time point Time

%%  UPDATE NEW-POTENTIALS  -------------------------------

update('NEW-POTENTIALS',T):-
   ((setof((T,T1,P),
           A^(TT^(TH^(M^(A cause 'POTEN'(TT+M,TH,P),
                         TT is T-M,
                         ismodel(TT,A),
                         eval(TH,T1))))),
            S1));
    S1=[]),
   ((setof((T,T1,P),
           A^(TH^(A cause 'POTEN'(T,TH,P),
                  ismodel(T,A),
                  eval(TH,T1))),
           S2));
    S2=[]),
   ((setof((T,T1,P),
           TT^(TH^(M^('POTEN'(TT+M,TH,P),
                      TT is T-M,
                      eval(TH,T1)))),
           S3));
    S3=[]),
   ((setof((T,T1,P),
           TH^('POTEN'(T,TH,P),
               eval(TH,T1)),
           S4));
    S4=[]),
   append_without_duplicates(S1,S2,SS1),
   append_without_duplicates(S3,S4,SS2),
   append_without_duplicates(SS1,SS2,Set),
   assertz('NEW-POTENTIALS'(T,Set)).

%%  UPDATE PARTIAL-CONSEQUENTS  --------------------------

update('PARTIAL-CONSEQUENTS',T):-
   ((setof(nec(T1,T,P),
           A^(TT^(TH^(M^(A cause nec(TH,TT+M,P),
                         TT is T-M,
                         ismodel(TT,A),
                         eval(TH,T1))))),
           S1),!);
    S1=[]),
```

```
   ((setof(nec(T1,T,P),
          A^(TH^(A cause nec(TH,T,P),
                 ismodel(T,A),
                 eval(TH,T1))),
          S2),!);
    S2=[]),
   ((setof(nec(T1,T,P),
          TT^(TH^(M^(nec(TH,TT+M,P),
                     TT is T-M,
                     eval(TH,T1)))),
          S3),!);
    S3=[]),
   ((setof(nec(T1,T,P),
          TH^(nec(TH,T,P),
              eval(TH,T1)),
          S4),!);
    S4=[]),
   append_without_duplicates(S1,S2,SS1),
   append_without_duplicates(S3,S4,SS2),
   append_without_duplicates(SS1,SS2,Set1),
   (
    (setof(nec(T,T,P),
          LHP^('NEW-POTENTIALS'(T,LHP),
               absmember((T,_,p-P),LHP),
               'PROJECT'(T,p-P,T,T,P)),
          Set2),!);
    Set2=[]
   ),
   append_without_duplicates(Set1,Set2,Set),
   assertz('PARTIAL-CONSEQUENTS'(T,Set)).

%% UPDATE NATURAL-DEATH ---------------------------------

update('NATURAL-DEATH',T):-
   (
    (setof(nec(T1,T,p-P),
          LP^('POTENTIALS'(T,LP),
              member((T1,T,p-P),LP)),
          Set),!);
    Set=[]
   ),
   assertz('NATURAL-DEATH'(T,Set)).

%% UPDATE CLIPPED ---------------------------------------

update('CLIPPED',T):-
   TT is T+1,
   (
    (setof(nec(T1,T,p-P),
          TT^(T3^(LP^(LPC^('POTENTIALS'(T,LP),
                           member((T1,_,p-P),LP),
                           'PROJECT'(T1,p-P,T3,TT,P),
                           'PARTIAL-CONSEQUENTS'(TT,LPC),
                           (member(nec(T3,TT, not P),LPC);
                            contradicts(nec(T3,TT,P),LPC))))))),
          Set),!);
    Set=[]
   ),
   assertz('CLIPPED'(T,Set)).

%% UPDATE POTENTIALS ------------------------------------

update('POTENTIALS',T):-
   TT is T-1,
   'NATURAL-DEATH'(TT,LHD),
   'CLIPPED'(TT,LC),
   append(LHD,LC,List1),
   (
    (setof((T1,TT,p-P),
          List1^member(nec(T1,TT,p-P),List1),
          Set1),!);
    Set1=[]
   ),
   'NEW-POTENTIALS'(T,LHP),
   'POTENTIALS'(TT,LP),
   append(LHP,LP,List2),
   difference_of_sets(List2,Set1,Set),
```

```
    assertz('POTENTIALS'(T,Set)).

%%  UPDATE CONSEQUENTS  ----------------------------------

update('CONSEQUENTS',T):-
    (
     (setof(nec(T3,T,P),
            T1^(T2^(LP^('POTENTIALS'(T,LP),
                        absmember((T1,T2,p-P),LP),
                        'PROJECT'(T1,p-P,T3,T,P)))),
            Set1),!);
     Set1=[]
    ),
    'PARTIAL-CONSEQUENTS'(T,LPC),
    append_without_duplicates(LPC,Set1,Set),
    assertz('CONSEQUENTS'(T,Set)).

% makemodel(+TimePoint)
%     assert the truth and falsity of the model at time TimePoint

makemodel(T):-
    'CONSEQUENTS'(T,S),
    tautologies(S).

% tautologies(+List)
%     given a List of sentences, assert those with positive prepositions
%     (i.e. those with no 'not's) as 'TRUE'/3 clauses and negative
%     prepositions (i.e. those with 'not's) as 'FALSE'/3 clases

tautologies([]):- !.
tautologies([nec(T1,T2, not P)|T]):-
    !,
    assertz('FALSE'(T1,T2,P)),
    tautologies(T).
tautologies([nec(T1,T2,P)|T]):-
    assertz('TRUE'(T1,T2,P)),
    tautologies(T).

% difference_of_sets(+Set1,+Set2,-Result)
%     given Set1 and Set2, returns as Result a set which is the difference
%     of Set1 and Set2, based on the chronological order defined by the
%     operator &<

difference_of_sets(L,[],L).
difference_of_sets(L1,[H|T],L):-
    remove_matches(H,L1,L2),
    difference_of_sets(L2,T,L).

% remove_matches(+Item,+List,-Result)
%     removes the Item from the List to return Result if the Item
%     succeed to entails/2 with the element in the List

remove_matches(_,[],[]).
remove_matches(A,[H|T],T):-
    entails(A,H),!.
remove_matches(A,[H|T],[H|R]):-
    remove_matches(A,T,R).

% entails(+Item1,+Item2)
%     succeeds if Item1 chronologically entails Item2

entails(A,A).
entails((T,T1,P),(T,T2,P)):-
    T1 &< T2.
entails((T,T1,P),(T,T2,P)):-
    T2 &< T1.

% eval(+Item,-Result)
%     returns evaluated value for Item, esp. inf for inf

eval(inf,inf):- !.
eval(T1,T2) :-
    T2 is T1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  SET HANDLING PREDICATES  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% absmember(+Tuple,+List)
%    succeeds if Tuple, or the negation of the proposition in the Tuple
%    is a member of List

absmember((T1,T2,p-P),List):-
   \+ (\+ P = not _),!,
   member((T1,T2,p-P),List).
absmember((T1,T2,p-not P),List):-
   member((T1,T2,p-P),List).

% ismodel(+TimePoint,+Conditions)
%    succeeds if Conditions are the model at TimePoint

ismodel(_,[]):- !.
ismodel(T,A):-
   integer(T),
   ismodel1(T,A).

ismodel1(T, nec(T+M,T+N, not A)):-
   !,
   T1 is T+M,
   T2 is T+N,
   'FALSE'(T1,T2,A).
ismodel1(T, nec(T+M,T+N,A)):-
   !,
   T1 is T+M,
   T2 is T+N,
   'TRUE'(T1,T2,A).
ismodel1(T, pos(T+M,T+N, not A)):-
   !,
   T1 is T+M,
   T2 is T+N,
   \+ 'TRUE'(T1,T2,A).
ismodel1(T, pos(T+M,T+N,A)):-
   !,
   T1 is T+M,
   T2 is T+N,
   \+ 'FALSE'(T1,T2,A).
ismodel1(T, nec(T+M,T+N, not A) and B):-
   !,
   T1 is T+M,
   T2 is T+N,
   'FALSE'(T1,T2,A),
   ismodel1(T,B).
ismodel1(T, nec(T+M,T+N,A) and B):-
   !,
   T1 is T+M,
   T2 is T+N,
   'TRUE'(T1,T2,A),
   ismodel1(T,B).
ismodel1(T, pos(T+M,T+N, not A) and B):-
   !,
   T1 is T+M,
   T2 is T+N,
   \+ 'TRUE'(T1,T2,A),
   ismodel1(T,B).
ismodel1(T, pos(T+M,T+N,A) and B):-
   !,
   T1 is T+M,
   T2 is T+N,
   \+ 'FALSE'(T1,T2,A),
   ismodel1(T,B).

ismodel1(T, nec(T-M,T-N, not A)):-
   !,
   T1 is T-M,
   T2 is T-N,
   'FALSE'(T1,T2,A).
ismodel1(T, nec(T-M,T-N,A)):-
   !,
   T1 is T-M,
   T2 is T-N,
   'TRUE'(T1,T2,A).
ismodel1(T, pos(T-M,T-N, not A)):-
   !,
   T1 is T-M,
```

```
    T2 is T-N,
    \+ 'TRUE'(T1,T2,A).
ismodel1(T, pos(T-M,T-N,A)):-
    !,
    T1 is T-M,
    T2 is T-N,
    \+ 'FALSE'(T1,T2,A).
ismodel1(T, nec(T-M,T-N, not A) and B):-
    !,
    T1 is T-M,
    T2 is T-N,
    'FALSE'(T1,T2,A),
    ismodel1(T,B).
ismodel1(T, nec(T-M,T-N,A) and B):-
    !,
    T1 is T-M,
    T2 is T-N,
    'TRUE'(T1,T2,A),
    ismodel1(T,B).
ismodel1(T, pos(T-M,T-N, not A) and B):-
    !,
    T1 is T-M,
    T2 is T-N,
    \+ 'TRUE'(T1,T2,A),
    ismodel1(T,B).
ismodel1(T, pos(T-M,T-N,A) and B):-
    !,
    T1 is T-M,
    T2 is T-N,
    \+ 'FALSE'(T1,T2,A),
    ismodel1(T,B).

ismodel1(T, nec(T-M,T, not A)):-
    !,
    T1 is T-M,
    'FALSE'(T1,T,A).
ismodel1(T, nec(T-M,T,A)):-
    !,
    T1 is T-M,
    'TRUE'(T1,T,A).
ismodel1(T, pos(T-M,T, not A)):-
    !,
    T1 is T-M,
    \+ 'TRUE'(T1,T,A).
ismodel1(T, pos(T-M,T,A)):-
    !,
    T1 is T-M,
    \+ 'FALSE'(T1,T,A).
ismodel1(T, nec(T-M,T, not A) and B):-
    !,
    T1 is T-M,
    'FALSE'(T1,T,A),
    ismodel1(T,B).
ismodel1(T, nec(T-M,T,A) and B):-
    !,
    T1 is T-M,
    'TRUE'(T1,T,A),
    ismodel1(T,B).
ismodel1(T, pos(T-M,T, not A) and B):-
    !,
    T1 is T-M,
    \+ 'TRUE'(T1,T,A),
    ismodel1(T,B).
ismodel1(T, pos(T-M,T,A) and B):-
    !,
    T1 is T-M,
    \+ 'FALSE'(T1,T,A),
    ismodel1(T,B).

ismodel1(T, nec(T,T+N, not A)):-
    !,
    T2 is T+N,
    'FALSE'(T,T2,A).
ismodel1(T, nec(T,T+N,A)):-
    !,
    T2 is T+N,
    'TRUE'(T,T2,A).
```

```
ismodel1(T, pos(T,T+1, not A)):-
   !,
   T2 is T+1,
   \+ 'TRUE'(T,T2,A).
ismodel1(T, pos(T,T+1,A)):-
   !,
   T2 is T+1,
   \+ 'FALSE'(T,T2,A).
ismodel1(T, nec(T,T+1, not A) and B):-
   !,
   T2 is T+1,
   'FALSE'(T,T2,A),
   ismodel1(T,B).
ismodel1(T, nec(T,T+1,A) and B):-
   !,
   T2 is T+1,
   'TRUE'(T,T2,A),
   ismodel1(T,B).
ismodel1(T, pos(T,T+1, not A) and B):-
   !,
   T2 is T+1,
   \+ 'TRUE'(T,T2,A),
   ismodel1(T,B).
ismodel1(T, pos(T,T+1,A) and B):-
   !,
   T2 is T+1,
   \+ 'FALSE'(T,T2,A),
   ismodel1(T,B).

ismodel1(T, nec(T,T, not A)):-
   !,
   'FALSE'(T,T,A).
ismodel1(T, nec(T,T,A)):-
   !,
   'TRUE'(T,T,A).
ismodel1(T, pos(T,T, not A)):-
   !,
   \+ 'TRUE'(T,T,A).
ismodel1(T, pos(T,T,A)):-
   !,
   \+ 'FALSE'(T,T,A).
ismodel1(T, nec(T,T, not A) and B):-
   !,
   'FALSE'(T,T,A),
   ismodel1(T,B).
ismodel1(T, nec(T,T,A) and B):-
   !,
   'TRUE'(T,T,A),
   ismodel1(T,B).
ismodel1(T, pos(T,T, not A) and B):-
   !,
   \+ 'TRUE'(T,T,A),
   ismodel1(T,B).
ismodel1(T, pos(T,T,A) and B):-
   !,
   \+ 'FALSE'(T,T,A),
   ismodel1(T,B).

% contradicts(+Term,+List)
%   succeeds if Term contradicts with any members of List

contradicts(A,LPC):-
   member(B,LPC),
   contra(A,B).

contra(nec(T1,T2,P),nec(T1,T2,Q)):-
   % ind(List) is a database where List contains sentences
   % which are mutually exclusive
   ind(M),
   member(P,M),
   member(Q,M).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  CLEAR UP ALL TEMPORARY CLAUSES  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc1:-
```

```
    retractall('NEW-POTENTIALS'(_,_)),
    retractall('PARTIAL-CONSEQUENTS'(_,_)),
    retractall('NATURAL-DEATH'(_,_)),
    retractall('CLIPPED'(_,_)),
    retractall('POTENTIALS'(_,_)),
    retractall('CONSEQUENTS'(_,_)),
    retractall('PROJECT'(_,_,_,_,_)),
    retractall('POTEN'(_,_,_)),
    retractall(_ cause _),
    retractall(nec(_,_,_)),
    retractall('TRUE'(_,_,_)),
    retractall('FALSE'(_,_,_)).
```

C.2 Program for computing bci sets

This is an implementation of the bci set construction described in Chapter 4. Similar to the cmi model construction, it involves the construction of the sets PRECONDITIONS, NEW-B-POTENTIALS, INITIATION, B-CLIPPED, B-POTENTIALS, PARTIAL-ANTECEDENTS and ANTECEDENTS. ANTECEDENTS conveys a bci set for given causal theory.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                            %%
%%          Abduction framework for logic CI                                  %%
%%                                                                            %%
%%          Coded by K. Nakata                                                %%
%%                                                                            %%
%%          March 1992                                                        %%
%%                                                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%
%%  DECLARATIONS  %%
%%%%%%%%%%%%%%%%%%%%%

:- dynamic 'PRECONDITIONS'/2,
           'ANTECEDENTS'/2,
           'PARTIAL-ANTECEDENTS'/2,
           'INITIATION'/2,
           'B-CLIPPED'/2,
           'B-POTENTIALS'/2,
           'NEW-B-POTENTIALS'/2,
           'TRUE'/3,
           'FALSE'/3,
           'PROJECT'/5,
           'B-PROJECT'/5,
           'POTEN'/3,
           cause/2,
           nec/3.

%%%%%%%%%%%%%%%%%%
%%  OPERATORS  %%
%%%%%%%%%%%%%%%%%%

:- op(500,fy,include).      % for file inclusion
:- op(300,fy,not).          % for negations
:- op(400,xfy,and).         % for conjunctions
:- op(500,xfx,cause).       % for "implications"
:- op(600,xfx,&<).          % for the comparison of time points

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  TOP-LEVEL PREDICATES  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Top-level  ---
%
% b_prove(+Theory,+Target,+InitialTime)
% b_prove(+Theory)
%    perform abduction for the causal theory in Theory for the Target
```

```
%    (if not specified, all) for the predicate Target (default=all)
%    between InitialTime (default=0) and the latest time point

b_prove(Theory):-
   b_prove(Theory,all,0).

b_prove(Theory,Target,InitialTime):-
   make_theory(Theory),
   ltp(TerminalTime),
   TT is TerminalTime + 1,
   assertz('B-POTENTIALS'(TT,[])),
   assertz('ANTECEDENTS'(TT,[])),
   update_sets(InitialTime,TT),
   StartTime is InitialTime+1,
   write('**** BCI SET FOR '),
   write(Theory),
   write(' ****'),nl,
   pretty_print(StartTime,TT,Target),
   clcl.

% ltp(-LTP)
%    finds the latest time point (LTP) of the given causal theory

ltp(LTP):-
   ltp(LTP,0),!.

ltp(LTP,Sofar):-
   nec(T1,inf,_),
   integer(T1),
   T is T1+1,              % if ltp is inf, then set ltp to be stp+1
   T > Sofar,
   ltp(LTP,T).
ltp(LTP,Sofar):-
   'POTEN'(T1,inf,_),
   integer(T1),
   T is T1+1,              % if ltp is inf, then set ltp to be stp+1
   T > Sofar,
   ltp(LTP,T).
ltp(LTP,Sofar):-
   nec(_,T,_),
   integer(T),
   T > Sofar,
   ltp(LTP,T).
ltp(LTP,Sofar):-
   'POTEN'(_,T,_),
   integer(T),
   T > Sofar,
   ltp(LTP,T).
ltp(LTP,LTP).

% update_sets(+InitialTime,+CurrentTime)
%    updates the sets PRECONDITIONS, NEW-B-POTENTIALS, INITIATION,
%    B-CLIPPED, B-POTENTIALS, PARTIAL-ANTECEDENTS, ANTECEDENTS in
%    order and recurses the process by decrementing the time step by
%    one until CurrentTime is equalised with InitialTime

update_sets(T,T):-
   !,nl,nl,nl.

update_sets(IT,TT):-
   NewTT is TT-1,
   nl,
   write('*** Updating Sets for Time-point '),
   write(NewTT),write(' ***'),nl,nl,
   update('PRECONDITIONS',NewTT),
   'PRECONDITIONS'(NewTT,PC),
   write('PRECONDITIONS                '),write(PC),nl,
   update('NEW-B-POTENTIALS',NewTT),
   'NEW-B-POTENTIALS'(NewTT,NPTN),
   write('NEW-B-POTENTIALS             '),write(NPTN),nl,
   update('INITIATION',TT),
   'INITIATION'(TT,INI),
   write('INITIATION                   '),write(INI),nl,
   update('B-CLIPPED',TT),
   'B-CLIPPED'(TT,CPD),
   write('B-CLIPPED                    '),write(CPD),nl,
   update('B-POTENTIALS',NewTT),
```

```
    'B-POTENTIALS'(NewTT,PTN),
    write('B-POTENTIALS              '),write(PTN),nl,
    update('PARTIAL-ANTECEDENTS',NewTT),
    'PARTIAL-ANTECEDENTS'(NewTT,PAN),
    write('PARTIAL-ANTECEDENTS       '),write(PAN),nl,
    update('ANTECEDENTS',NewTT),
    'ANTECEDENTS'(NewTT,ANT),
    write('ANTECEDENTS               '),write(ANT),nl,
    makeset(NewTT),
    nl,
    update_sets(IT,NewTT).

% pretty_print(+InitialTime,+TerminalTime,+Target)
%    prints out the tautologies for the predicates specified in the list
%    Target; Target = all for all events

pretty_print(T,T,L):-
    !,nl,
    (
     (T >9, write('  t= '));
     (T=<9, write('  t=  '))
    ),
    write(T),write(' :    '),
    print_truth(T,L),
    nl,nl.
pretty_print(IT,TT,all):-
    !,nl,
    (
     (IT >9, write('  t= '));
     (IT=<9, write('  t=  '))
    ),
    write(IT),write(' :    '),
    print_truth(IT,all),
    ITT is IT+1,
    pretty_print(ITT,TT,all).
pretty_print(IT,TT,L):-
    !,nl,
    (
     (IT >9, write('  t= '));
     (IT=<9, write('  t=  '))
    ),
    write(IT),write(' :    '),
    print_truth(IT,L),
    ITT is IT+1,
    pretty_print(ITT,TT,L).

print_truth(T,all):-
    setof(P,T1^'TRUE'(T1,T,P),S),!,
    form_print(S).
print_truth(T,L):-
    \+ L == all,
    setof(P,T1^(L^('TRUE'(T1,T,P),member(P,L))),S),!,
    form_print(S).
print_truth(_,_):-
    write('---'),nl.

form_print([]):- !.
form_print([H|T]):-
    write(H),nl,tab(13),
    form_print(T).

%%%%%%%%%%%%%%%%%%%%%%
%%  UPDATING SETS  %%
%%%%%%%%%%%%%%%%%%%%%%

% update(+Set,+Time)
%    updates the Set at the time point Time

%% UPDATE PRECONDITIONS ------------------------------

update('PRECONDITIONS',T):-
    TT is T+1,
    % Collecting all the known facts
    (
(setof(nec(T,T,P),
       TS^(TE^(nec(TS,TE,P),
       TS @< T,
```

```
      T &< TE)),
      S),!);
 S=[]
    ),
    % Collecting possible events on the rise of the facts above
    setof(#L1-S1,
  (A1^(
       (setof(nec(T,T,P1),
      (A1 cause nec(T+1,_,PP),
       eval(T+1,TT),
       isset(TT,nec(TT,TT,PP)),
       condition(nec(T,T,P1),A1)),
      S1),
n_intersection(S1,S,IS1,#1),
#1 > 0,
ante_fulfill(S1,S),!);
       (S1=[], #1=0)
   ),
  length(S1,##1),
  #L1 is ##1-#1),
  SS1),
    setof(#L2-S2,
  (A2^(
       (setof(nec(T,T,P2),
      (A2 cause 'POTE#'(T+1,_,p-PP2),
       eval(T+1,TT),
       isset(TT,nec(TT,TT,PP2)),
       condition(nec(T,T,P2),A2)),
      S2),
n_intersection(S2,S,IS2,#2),
#2 > 0,
ante_fulfill(S2,S),!);
       (S2=[], #2=0)
   ),
   length(S2,##2),
   #L2 is ##2-#2),
  SS2),
    append_without_duplicates(SS1,SS2,SS3),
    remove(0-[],SS3,SS4),
    sort_by_priority(SS4,L),
    append_without_duplicates(S,L,Set),
    assertz('PRECONDITIONS'(T,Set)).

%% NEW-B-POTENTIALS ---------------------------------

update('NEW-B-POTENTIALS',T):-
    TT is T+1,
    'PRECONDITIONS'(T,S1),
    (
(setof((T1,T2,p-P1),
       (member(nec(TI,T2,P1),S1),
'POTE#'(T1,T3,p-P1),
T1 &< TI,
T2 &< T3),
       SS1),!);
SS1=[]
    ),
    (
(setof((I,T5,p-P2),
       A^(member(nec(TJ,T5,P2),S1),
A cause 'POTE#'(T#+1,T6,p-P2),
T5 &< T6),
       SS2),!);
SS2=[]
    ),
    (
(setof((T7,T8,p-P3),
       (isset(TT,nec(TK,T8,P3)),
'POTE#'(T7,T9,p-P3),
T7 &< TK,
T8 &< T9),
       SS3),!);
SS3=[]
    ),
    (
(setof((I,T12,p-P4),
       A^(isset(TT,nec(TL,T12,P4)),
```

```
A cause 'POTEN'(TN+1,T13,p-P4),
T12 &< T13),
        SS4),!);
SS4=[]
    ),
    append_without_duplicates(SS1,SS2,Set1),
    append_without_duplicates(SS3,SS4,Set2),
    append_without_duplicates(Set1,Set2,Set),
    assertz('NEW-B-POTENTIALS'(T,Set)).

%% UPDATE INITIATION ----------------------------

update('INITIATION',T):-
    'B-POTENTIALS'(T,S1),
    (
(setof(nec(T,T1,p-P),
       member((T,T1,p-P),S1),
       Set),!);
Set=[]
    ),
    assertz('INITIATION'(T,Set)).

%% UPDATE B-CLIPPED -----------------------------------

update('B-CLIPPED',T):-
    TT is T-1,
   'B-POTENTIALS'(T,S1),
   'PRECONDITIONS'(TT,S2),
   'ANTECEDENTS'(T,S3),
    (
(setof(nec(T,T3,p-P1),
       (member((T1,T3,p-P1),S1),
'B-PROJECT'(TT,p-P1,T2,T3,P1),
member(nec(TT,T2,not P1),S2)),
       Set1),!);
Set1=[]
    ),
    (
(setof(nec(T,T6,p-not P2),
       (member((T4,T6,p-not P2),S1),
'B-PROJECT'(TT,p-not P2,T5,T6,not P2),
member(nec(TT,T5,P2),S2)),
       Set2),!);
Set2=[]
    ),
    (
(setof(nec(T,T9,p-P3),
       (member((T7,T9,p-P3),S1),
'B-PROJECT'(TT,p-P3,T8,T9,P3),
member(nec(TT,T8,not P3),S3)),
       Set1),!);
Set3=[]
    ),
    (
(setof(nec(T,T12,p-not P4),
       (member((T10,T12,p-not P4),S1),
'B-PROJECT'(TT,p-not P4,T11,T12,not P4),
member(nec(TT,T11,P4),S3)),
       Set2),!);
Set4=[]
    ),
    (
(setof(nec(T,T15,p-P5),
       (member((T13,T15,p-P5),S1),
'B-PROJECT'(TT,p-P5,T14,T15,P5),!,
contradicts(nec(TT,T14,P5),S2)),
       Set5),!);
Set5=[]
    ),
    (
(setof(nec(T,T18,p-P6),
       (member((T16,T18,p-P6),S1),
'B-PROJECT'(TT,p-P6,T17,T18,P6),!,
contradicts(nec(TT,T17,P6),S3)),
       Set6),!);
Set6=[]
    ),
```

```
    append_without_duplicates(Set1,Set2,SS1),
    append_without_duplicates(Set3,Set4,SS2),
    append_without_duplicates(Set5,Set6,SS3),
    append_without_duplicates(SS1,SS2,SS4),
    append_without_duplicates(SS3,SS4,Set),
    assertz('B-CLIPPED'(T,Set)).

%% UPDATE B-POTENTIALS ----------------------------------------

update('B-POTENTIALS',T):-
   TT is T+1,
   'B-POTENTIALS'(TT,S1),
   'NEW-B-POTENTIALS'(T,S2),
   append_without_duplicates(S1,S2,SS),
   'INITIATION'(TT,S3),
   'B-CLIPPED'(TT,S4),
    append_without_duplicates(S3,S4,L),
    (
(setof((_,_,p-P),
       member(nec(TT,T1,p-P),L),
       S5),!);
S5=[]
    ),
    (
(setof((T1,T2,p-P),
       (member((T1,T2,p-P),SS),
\+ member((_,_,p-P),S5)),
       Set),!);
Set=[]
    ),
   assertz('B-POTENTIALS'(T,Set)).

%% UPDATE PARTIAL-ANTECEDENTS ---------------------------------

update('PARTIAL-ANTECEDENTS',T):-
    TT is T+1,
    'B-POTENTIALS'(T,S1),
    'PRECONDITIONS'(T,S2),
    'INITIATION'(TT,S3),
    (
(setof(nec(T,T,PP1),
       (A1 cause nec(_,T1+N1,P1),
isset(TT,nec(_,TT,P1)),
\+ member((_,TT,p-P1),S1),
condition(nec(_,TT,PP1),A1),
eval(T1,TT),
\+ member((_,TT,p-not PP1),S1),
\+ member(nec(_,TT,not PP1),S2),
\+ contradicts(nec(_,TT,PP1),S2)),
       Set1),!);
Set1 = []
    ),
    (
(setof(nec(T,T,PP2),
       (A2 cause 'POTEN'(T2+N2,_,p-P2),
member(nec(TT,_,p-P2),S3),
condition(nec(TT,_,PP2),A2),
eval(T2,TT),
\+ member((_,TT,p-not PP2),S1),
\+ member(nec(_,TT,not PP2),S2),
\+ contradicts(nec(_,TT,PP2),S2)),
       Set2),!);
Set2 = []
    ),
   append_without_duplicates(Set1,Set2,SS),
   append_without_duplicates(SS,S2,Set),
   assertz('PARTIAL-ANTECEDENTS'(T,Set)).

%% UPDATE ANTECEDENTS -----------------------------------

update('ANTECEDENTS',T):-
   'PARTIAL-ANTECEDENTS'(T,S1),
   'B-POTENTIALS'(T,S2),
    (
(setof(nec(T,T,P),
       T1^(T3^(member((T1,T3,p-P),S2),
       'B-PROJECT'(T,p-P,T2,T3,P))),
```

```
        S3),!).
S3 = []),
   append_without_duplicates(S1,S3,Set),
   assertz('ANTECEDENTS'(T,Set)).

% ante_fulfill(+Set,+Conditions)
%    returns true if Set does not contain any element which is
%    contradictory to Conditions

ante_fulfill([],_):- !.
ante_fulfill([H|T],S):-
    \+ contradicts(H,S),
    ante_fulfill(T,S).

% sort_by_priority(+List,-SortedList)
%    given a List consisting of (possibly empty) indexed elements,
%    returns as SortedList the first element in the sorted list
%    without the index

sort_by_priority([],[]):-!.
sort_by_priority(S,L):-
   sort(S,[_-L|_]).

% condition(+Term,+Condition)
%   is TRUE if Term is a necessary (box) condition among Condition

condition(Term,Condition):-
   boxes(Condition,Boxes),
   member(Term,Boxes).

% makeset(+TimePoint)
%    assert the truth and falsity of the model at time TimePoint

makeset(T):-
   'ANTECEDENTS'(T,S),
   tautologies(S).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  SET HANDLING PREDICATES  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% absmember(+Tuple,+List)
%    succeeds if Tuple, or the negation of the proposition in the Tuple
%    is a member of List

absmember((T1,T2,p-P),List):-
   \+ (\+ P = not _),!,
   member((T1,T2,p-P),List).
absmember((T1,T2,p-not P),List):-
   member((T1,T2,p-P),List).

% isset(+TimePoint,+Term)
%    succeeds if Term is included in the time bounded set at TimePoint

isset(T,nec(T1,T2,not P)):-
    'FALSE'(T,T,P),
    T &< T1,
    T2 &< T.
isset(T,nec(T1,T2,P)):-
    'TRUE'(T,T,P),
    T &< T1,
    T2 &< T.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  CLEAR UP ALL TEMPORARY CLAUSES   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clcl:-
   retractall('PRECONDITIONS'(_,_)),
   retractall('ANTECEDENTS'(_,_)),
   retractall('PARTIAL-ANTECEDENTS'(_,_)),
   retractall('INITIATION'(_,_)),
   retractall('B-CLIPPED'(_,_)),
   retractall('B-POTENTIALS'(_,_)),
   retractall('NEW-B-POTENTIALS'(_,_)),
   retractall('PROJECT'(_,_,_,_,_,_)),
   retractall('B-PROJECT'(_,_,_,_,_,_)),
```

```
   retractall('POTEN'(_,_,_)),
   retractall(_ cause _),
   retractall(nec(_,_,_)),
   retractall('TRUE'(_,_,_)),
   retractall('FALSE'(_,_,_)).
```

C.3 Program for bidirectional sweeping

This is an implementation of the bidirectional sweeping described in Chapter 4. Once the cmi model and a bci for a causal theory is computed in a list as `cmi_model/1` and `bci_set/1`, `find_event` identifies the discrepancies and suggests candidate events to have occurred to obtain a reconciliation set.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%%          Bidirectional sweeping module                                       %%
%%                                                                              %%
%%          Coded by K. Nakata                                                  %%
%%                                                                              %%
%%          January 1993                                                        %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TOP-LEVEL PREDICATES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Top-level ---
%
% find_event(+InitialTime,-List,-Events)
% find_event
%    by comparing the forward and backward sweeping over a causal
%    theory, finds the List of discrepancies and suggests the missing
%    Events which should have taken place.

find_event:-
   find_event(0,L,E).

find_event(InitialTime,Disc,Events):-
   cmi_model(CMI),
   bci_set(BCI),
   disagree(CMI,BCI,InitialTime,Disc),
   analyse_disc(Disc,Events).

% disagree(+Set1,+Set2,+StartTime,-Discrepancies)
%    finds the Discrepancies between the sets Set1 and Set2 at various
%    time points starting from StartTime.

disagree(CMI,BCI,InitialTime,Disc):-
   disagree(CMI,BCI,InitialTime,Disc,[]).

disagree([],[],_,Disc,Disc):-!.
disagree(CMI,BCI,Time,Disc,DiscSofar):-
   TT is Time+1,
   time_events(TT,CMI,EventC,CMIRest),
   time_events(TT,BCI,EventB,BCIRest),
   diff_events(EventC,EventB,DiffEvent),
   disagree(CMIRest,BCIRest,TT,Disc,[TT-DiffEvent|DiscSofar]).

% time_events(+Time,+SetOfEvents,-Events,-RestOfEvents)
%    given SetOfEvents, provides Events at Time, and returns the
%    remainder as RestOfEvents.

time_events(T,List,Events,Rest):-
   time_events(T,List,Events,[],Rest,[]).

time_events(_,[],Events,Events,Rest,Rest):-!.
time_events(T,['TRUE'(T,T,P)|R],Events,Sofar,Rest,RestSofar):-
   time_events(T,R,Events,[P|Sofar],Rest,RestSofar),!.
time_events(T,['FALSE'(T,T,P)|R],Events,Sofar,Rest,RestSofar):-
```

```
    time_events(T,R,Events,[n-P|Sofar],Rest,RestSofar),!.
time_events(T,[H|R],Events,Sofar,Rest,RestSofar):-
    time_events(T,R,Events,Sofar,Rest,[H|RestSofar]).

% diff_events(+EventList1,+EventList2,-Discrepancies)
%     Discrepancies is the list of contradicting events between
%     EventList1 and EventList2.  The elements of Discrepancies are
%     described in the following way:
%     O@Term          Term appears in EventList2 but not in EventList1
%     Term@O          Term appears in EventList1 but not in EventList2
%     Term@n-Term     the negation of Term in EventList1 appears in
%                     EventList2
%     n-Term@Term     the negation of Term in EventList2 appears in
%                     EventList1
%     Term1@Term2     Term1 in EventList1 contradicts with Term2 in
%                     EventList2

diff_events([],[],[]):- !.
diff_events([],[H|T],[(O@H)|Diff]):- !,
    diff_events([],T,Diff).
diff_events([H|T],List2,Diff):-
    member(H,List2),!,
    remove(H,List2,L2),
    diff_events(T,L2,Diff).
diff_events([n-H|T],List2,[(n-H@H)|Diff]):-
    member(H,List2),!,
    remove(H,List2,L2),
    diff_events(T,L2,Diff).
diff_events([H1|T],List2,[(H1@n-H1)|Diff]):-
    member(n-H1,List2),!,
    remove(n-H1,List2,L2),
    diff_events(T,L2,Diff).
diff_events([H1|T],List2,[(H1@H2)|Diff]):-
    ind(M),
    member(H1,M),
    member(H2,List2),
    \+ H1==H2,
    member(H2,M),!,
    remove(H2,List2,L2),
    diff_events(T,L2,Diff).
diff_events([H|T],List2,[(H@O)|Diff]):-
    \+ member(H,List2),
    diff_events(T,List2,Diff).

% analyse_events(+Discrepancies,-PotentialEvents)
%     given the list of Discrepancies for all time points (tagged with
%     Time- info), print it out and return PotentialEvents as
%     candidates of events which resolves the discrepancies.

analyse_disc(Disc,Events):-
    sort(Disc,DiscS),
    print_disc(DiscS),
    identify_events(DiscS,[],Events),
    print_events(Events).

% identify_events(+Discrepancies,+Sofar,-Events)
%     returns candidates of events which resolves the discrepancies in
%     Discrepancies

identify_events([],Events,Events) :- !.
identify_events([_-[]|R],Sofar,Events):-
    identify_events(R,Sofar,Events).
identify_events([T-[O@Term|R]|Rest],Sofar,Events):- !,
    identify_events([T-R|Rest],[nec(T,T,Term)|Sofar],Events).
identify_events([T-[Term@O|R]|Rest],Sofar,Events):- !,
    identify_events([T-R|Rest],[nec(T,T,Term)|Sofar],Events).
identify_events([T-[n-T1@n-T2|R]|Rest],Sofar,Events):-
    A cause nec(TT+M,_,not T2),
    condition(nec(T,T,not T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[n-T1@n-T2|R]|Rest],Sofar,Events):-
    A cause 'POTEN'(TT+M,_,p-not T2),
    condition(nec(T,T,not T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
```

```
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[n-T1@T2|R]|Rest],Sofar,Events):-
    A cause nec(TT+M,_,T2),
    condition(nec(T,T,not T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[n-T1@T2|R]|Rest],Sofar,Events):-
    A cause 'POTEN'(TT+M,_,p-T2),
    condition(nec(T,T,not T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[T1@n-T2|R]|Rest],Sofar,Events):-
    A cause nec(TT+M,_,not T2),
    condition(nec(T,T,T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[T1@T2|R]|Rest],Sofar,Events):-
    A cause nec(TT+M,_,T2),
    condition(nec(T,T,T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[T1@T2|R]|Rest],Sofar,Events):-
    A cause 'POTEN'(TT+M,_,p-T2),
    condition(nec(T,T,T1),A),!,
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).

identify_events([T-[n-T1@n-T2|R]|Rest],Sofar,Events):- !,
    A cause nec(TT+M,_,not T2),
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[n-T1@n-T2|R]|Rest],Sofar,Events):- !,
    A cause 'POTEN'(TT+M,_,p-not T2),
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[n-T1@T2|R]|Rest],Sofar,Events):- !,
    A cause nec(TT+M,_,T2),
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[n-T1@T2|R]|Rest],Sofar,Events):- !,
    A cause 'POTEN'(TT+M,_,p-T2),
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,not T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[T1@n-T2|R]|Rest],Sofar,Events):- !,
    A cause nec(TT+M,_,not T2),
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[T1@T2|R]|Rest],Sofar,Events):- !,
    A cause nec(TT+M,_,T2),
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).
identify_events([T-[T1@T2|R]|Rest],Sofar,Events):- !,
    A cause 'POTEN'(TT+M,_,p-T2),
```

```
    setof(nec(T,T,Term),condition(nec(T,T,Term),A),S),
    remove(nec(T,T,T1),S,SS),
    append(SS,Sofar,NewEvents),
    identify_events([T-R|Rest],NewEvents,Events).

% print_disc(+Discrepancies)
%    pretty-prints the contents of Discrepancies.

print_disc(L):-
    nl,
    print(' **********   Discrepancies   ********** '),nl,
    print('Time point       forward          backward    '),nl,
    print('-------------------------------------------'),nl,
    print_d(L).

print_d([]):- !,
    print(' ********   end of description   ********* ').
print_d([_-[]|R]):- !,
    print_d(R).
print_d([T-[O@n-Term|R]|Rest]):- !,
    print('    '), print(T), print('                ----           '),
    print('not '), print(Term),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[n-Term@O|R]|Rest]):- !,
    print('    '), print(T), print('          '), print('not '),
    print(Term), print('            ----'),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[O@Term|R]|Rest]):- !,
    print('    '), print(T), print('                ----           '),
    print(Term),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[Term@O|R]|Rest]):- !,
    print('    '), print(T), print('          '), print(Term),
    print('            ----'),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[n-T1@n-T2|R]|Rest]):- !,
    print('    '), print(T), print('              '), print('not '),
    print(T1), print('          '), print('not '), print(T2),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[T1@n-T2|R]|Rest]):- !,
    print('    '), print(T), print('              '),
    print(T1), print('          '), print('not '), print(T2),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[n-T1@T2|R]|Rest]):- !,
    print('    '), print(T), print('              '), print('not '),
    print(T1), print('          '), print(T2),nl,
    print_r(R),
    print_d(Rest).
print_d([T-[T1@T2|R]|Rest]):- !,
    print('    '), print(T), print('              '), print(T1),
    print('          '), print(T2),nl,
    print_r(R),
    print_d(Rest).

print_r([]):- !.
print_r([O@n-Term|R]):- !,
    print('                '), print('          ----          '),
    print('not '),print(Term),nl,
    print_r(R).
print_r([n-Term@O|R]):- !,
    print('                '), print('not '), print(Term),
    print('          ----'),nl,
    print_r(R).
print_r([O@Term|R]):- !,
    print('                '), print('          ----          '),
    print(Term),nl,
    print_r(R).
print_r([Term@O|R]):- !,
    print('                '), print(Term), print('          ----'),nl,
    print_r(R).
print_r([n-T1@n-T2|R]):-
```

```
    print('              '), print('        '), print('not '),
    print(T1), print('      '), print('not '), print(T2),nl,
    print_r(R).
print_r([T1@n-T2|R]):-
    print('              '), print('        '),
    print(T1), print('      '), print('not '), print(T2),nl,
    print_r(R).
print_r([n-T1@T2|R]):-
    print('              '), print('        '), print('not '),
    print(T1), print('      '), print(T2),nl,
    print_r(R).
print_r([T1@T2|R]):-
    print('              '), print('        '), print(T1),
    print('     '), print(T2),nl,
    print_r(R).

% print_events(+Events)
%    pretty-prints the contents of Events.

print_events(L):-
    sort(L,L1),
    nl,nl,
    print(' *** The following event(s) should have occurred ***'),nl,
    print_e(L1).

print_e([]):- !.
print_e([nec(T,T,n-P)|R]):-
    print('At time point '), print(T), print('  not '), print(P), nl,
    print_e(R).
print_e([nec(T,T,P)|R]):-
    print('At time point '), print(T), print('  '), print(P), nl,
    print_e(R).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CLEAR UP ALL TEMPORARY CLAUSES   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clcl:-
    retractall(_ cause _),
    retractall(nec(_,_,_)),
    retractall('TRUE'(_,_,_)),
    retractall('FALSE'(_,_,_)).
```

C.4 Programs for design modules

This is an implementation of the proposed design modules described in Chapter 7. There are four modules involved: model construction module, discrepancy detection module, rule construction module and rule proving module. In this implementation, the cmi model construction is used as model construction module.

Top-level control

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%                Behaviour-oriented design modules                       %%
%%                                                                        %%
%%                 -- top level control program --                        %%
%%                                                                        %%
%%                Coded by K. Nakata                                      %%
%%                                                                        %%
%%                August 1993                                             %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% go/0
%    Top level predicate to start up interactive design process.
```

```
%   It requires following predicates:
%       MODEL CONSTRUCTOR     --- prove/2 (cmi model constructor)
%       DISCREPANCY DETECTOR --- disc/4
%       NEW RULE SUGGESTION   --- newrule/3
%       RULE PROVER           --- proof/2

go :-
   nl,nl,
   input_user(theory,Theory),
   input_user(specification,Spec),
   input_user(timespan,T),!,
   go(Theory,T,Spec).

go(Theory,T,Spec):-
   nl,
   write('Constructing cmi model...'),
   nl,
   %%%%%%%%%%%%%%%%%%%%%%%%%%%% Creating CMI model
   prove(Theory,T),!,
   %%%%%%%%%%%%%%%%%%%%%%%%%%%% Comparing the prediction and the spec
   make_spec_list(Spec,SpecList),
   disc(SpecList,0,0,D),
   (D == no_discrepancy ->
    (write('No discrepancy detected'),nl,nl);
    (D=(TP,E),
     write('Discrepancy occurring at time point '),
     write(TP),nl,tab(4),
     write(E),
     write(' should be obtained at this point.'),nl,nl,
     write('Suggesting new rule to add and prove... '),nl,nl,
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Suggesting new rules
     preceding_events(TP,SetOfEvents),
     newrule(SetOfEvents,E,NewRule),
     write(NewRule),nl,
     nl,input_yn('Rule',YN1),nl,
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Proving new rules
     (YN1 == q -> (true);
(proof(NewRule,SetOfEvents,ProofRule),
 nl,input_yn('Proof',YN2),nl,
 (YN2 == q -> true))))),
     clcl.

% input_user(+Type,+Input)
%    Reads Input from the std input according to the Type of the input.
%    Types: theory (causal theory)
%           specification (behavioural specification)
%           timespan (defines the span of prediction for c.m.i. model)

input_user(Type,Input):-
  ((Type == theory,
    prompt(X,'>> Enter causal theory file name: '));
   (Type == specification,
    prompt(X,'>> Enter specification file name: '));
   (Type == timespan,
    prompt(X,'>> Enter duration of prediction: '))),
   repeat,
   read(Input),nl,
   check_input(Input,Type),
   prompt(_,X),
   nl.

% check_input(+Input,+Type)
%   returns true if Input is appropriate for given Type

check_input(Input,timespan):- !,
   check_integer(Input).
check_input(Input,Type):-
   check_file_exist(Input,Type).

check_file_exist(File,Type):-
   name(File,L1),
   name('test -f ',Test),
   append(Test,L1,L2),
   name(LsFile,L2),
   unix(shell(LsFile,Status)),
   (Status =\= 0 ->                          % tests if the file exists
      (write(Type),
```

```
        write(' file: '),                          % if not, ask for another
        write(File),                               % file.
        write(' does not exist'),nl,
        prompt(_,'Type in another file name: '),
        fail);
       true).

check_integer(I):-
    integer(I) ->
       true;
       (write(I),
        write(' is not an integer.'),nl,
        prompt(_,'Type in an integer: '),
        fail).

% input_yn(+Type)
%    Retrieve user inputs, y(yes),n(no),p(subproof),q(quit) according
%    to the Type.  "subproof" provides the input for the proof/3
%    predicate from that point. "quit" aborts the process.

input_yn(Type):-
    write(Type),write('--->'),nl,
    prompt(X,'>> Another candidate?[y/n/p/q]: '),
    read(YN),nl,
    (YN == y -> fail;
     (YN == n -> prompt(_,X);
      (YN == p -> subproof;
       (YN == q -> (prompt(_,X),clcl,true);
        (nl,write('Please answer y or n.'),nl,
         input_yn(Type)))))).

% input_yn(+Type,-Reply)
%    Retrieve user inputs, y(yes),n(no),p(subproof),q(quit) according
%    to the Type.  "subproof" provides the input for the proof/3
%    predicate from that point. "quit" aborts the process.

input_yn(Type,YN):-
    write(Type),write('--->'),nl,
    prompt(X,'>> Another candidate?[y/n/p/q]: '),
    read(YN),nl,
    (YN == y -> fail;
     (YN == n -> prompt(_,X);
      (YN == p -> subproof;
       (YN == q -> (prompt(_,X),clcl,true);
        (nl,write('Please answer y or n.'),nl,
         input_yn(Type)))))).

% subproof
%   commences a proof subprocess for a certain clause
%   (input interactively)

subproof:-
    write('*** Proof subprocess ON ***'),nl,
    prompt(_,'>> Clause: '),
    read(C),
    C =.. [_,_,_,Event],
    CC = nec(t+1,t+1,Event),
    prompt(_,'>> At time: '),
    read(T),
    preceding_events(T,E),
    newrule(E,CC,NewRule),
    write(NewRule),nl,
    nl,input_yn('Rule'),nl,
    proof(NewRule,E,ProofRule),
    nl,input_yn('Proof'),nl,
    write('*** Proof subprocess OFF ***'),nl.
```

Discrepancy detection module

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                         %%
%%    Discrepancy checking module          %%
%%                                         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% TOP LEVEL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% disc(+SpecList,+BeginTime,-Discrepancy)
%    SpecList is the list of events in the specification which are
%    sorted in temporal order (done in the beginning).  BeginTime specifies
%    the time point where the search should begin.  Discrepancy is the
%    first diverging point in the histories.

disc([],_,_,no_discrepancy):- !.              % Termination => no discrepancy
disc([(T,E)|R],PrevT,BT,D):-
   T > PrevT,!,
   TT is BT+1,
   (agree_check(T,E,TT,BT) ->
    disc(R,T,BT,D);
    D = (TT,E)).
disc([(T,E)|R],T,BT,D):- !,
   ('TRUE'(BT,_,E) ->
    disc(R,T,BT,D);
    D = (BT,E)).

% agree_check(+SpecTime,+SpecEvent,+BeginTime,-NewTime)
%   sets NewTime to the appropriate time point where the comparison
%   should commence.

agree_check(_,not E,BT,T):- !,
   'TRUE'(T,_,E),
   T >=BT,
   fail.
agree_check(_,E,BT,T):-
   'FALSE'(T,_,E),!,
   T >= BT,
   fail.
agree_check(_,E,BT,T):-
   'TRUE'(T,_,not E),!,
   T >= BT,
   fail.
agree_check(_,E,BT,T):-
   'TRUE'(T,_,E),
   T >= BT.


%% Additional code to create Specification List from the specification
%% data file.
% make_spec_list(+SpecFile,-SpecList)
%   reads SpecFile and creates SpecList, which is a chronologically
%   sorted list of specified behaviour.

make_spec_list(SpecFile,SpecList):-
   name(SpecFile,L1),
   name(LsFile,[116,101,115,116,32,45,102,32|L1]),
   unix(shell(LsFile,Status)),
   (Status =\= 0 ->                           % tests if the file exists
      (write('Specification file: '),         %  if not, ask for another
       write(SpecFile),                       %  file.
       write(' does not exist'),nl,
       prompt(X,'Type in another file name: '),
       read(NewFile),
       prompt(_,X),
       make_spec_list(NewFile,SpecList));
      (open(SpecFile,read,Stream),
       process_spec(Stream,[],SpecList),
       close(Stream))).


% process_spec(+Stream,+Sofar,-SpecList)
%    read from Stream (a file) the specification and make them into
%    a list of events (SpecList)

process_spec(Stream,Sofar,SpecList):-
   read(Stream,Term),
   (Term == end_of_file ->
      chron_sort(Sofar,SpecList);
      (process_term(Term,TermP),
       append(TermP,Sofar,NewSofar),
       process_spec(Stream,NewSofar,SpecList))).
```

```
% process_term(+Term,-Processed)
%    process (InitTime,EndTime,Event) form specification to
%    (Time,Event) specification

process_term(Term,TermP):-
   process_term1(Term,[],TermP).

process_term1((T,T,P),Sofar,[(T,P)|Sofar]):- !.
process_term1((T1,T2,P),Sofar,TermP):-
   TT1 is T1 + 1,
   process_term1((TT1,T2,P),[(T1,P)|Sofar],TermP).

% chron_sort(+List,-SortedList)
%    using built-in sort/2

chron_sort(List,SortedList):-
   sort(List,SortedList).
```

Rule construction module

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                              %%
%%          Making New Rules                    %%
%%                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%
%% OPERATORS  %%
%%%%%%%%%%%%%%%%%%

%:- op(300,fy,not).          % for negations
%:- op(400,xfy,and).         % for conjunctions
%:- op(500,xfx,cause).       % for "implications"

% newrule(+SetOfEvents,+Consequent,-NewRule)
%    Creates all possible NewRules from given SetOfEvents and the
%    Consequent

newrule([],Consequent,Consequent).
newrule(Event,Consequent,ConEvent cause TConsequent):-
   attach_time(Event,Consequent,TEvent,TConsequent),
   subset(TEvent,E),
   newrule1(E,ConEvent).

newrule1([Event],Event).
newrule1([E1|Events],E1 and R):-
   newrule1(Events,R).

% attach_time(+Event,+Consequent,-TEvent,-TConsequent)
%    TEvent is of the form nec(t,t,E) where E is an element of E;
%    TConsequent is of the form nec(t+1,t+1,Consequent)

attach_time([],C,[],nec(t+1,t+1,C)).
attach_time([H|T],C,[nec(t,t,H)|TE],TC):-
   attach_time(T,C,TE,TC).

% subset(+Set,-Subset)
%    Set and its Subsets

subset(Set,Subset):-
   length(Set,L),
   subset1(Set,1,L,Subset).

subset1(Set,L,L,Set):- !.
subset1(Set,N,L,Subset):-
   N < L,
   combination(Set,N,Subset).
subset1(Set,N,L,Subset):-
   N1 is N+1,
   subset1(Set,N1,L,Subset).

% combination(+List,+Number,-Combination)
%    Takes out Number elements from the List and finds all possible
%    Combination by backtracking
```

```
combination(_,0,_):- fail.
combination([H|_],1,[H]).
combination([_|T],N,Comb):-
   combination(T,N,Comb).
combination([H|T],N,[H|Comb]):-
   N1 is N-1,
   combination(T,N1,Comb).

% preceding_events(+TimePoint,-SetOfEvents)
%   collects SetOfEvents prior to the TimePoint

preceding_events(T,Events):-
   TT is T-1,
   setof(E,'TRUE'(TT,_,E),Events),!.
preceding_events(_,[]).
```

Rule proving module

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                              %%
%%        Proving New Rules                     %%
%%                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%
%%  OPERATORS   %%
%%%%%%%%%%%%%%%%%%

:- op(300,fy,not).          % for negations
:- op(400,xfy,and).         % for conjunctions
:- op(500,xfx,cause).       % for "implications"

%%%%%%%%%%%%%%%%%%
%%  TOP LEVEL   %%
%%%%%%%%%%%%%%%%%%

% proof(+Rule,+EventList,-ProveRule)
%   Given Rule, constructs ProveRule by performing abductive operation
%   for the consequent and finding alternative rules which fulfills
%   the antecedents.

proof(P cause Q,EventList,ProveRule):-
   abduction(Q,Results),
   exp(P,EP,AndProve),
   next_step(EP,Q,Results,ProveRule),
   valid_cause(ProveRule),
   valid_condition(AndProve,EventList,NewProve,Flag),
   \+(Flag == 1),
   nl,
   write('Assert new rule:'),nl,
   tab(3),
   write(ProveRule),
   print_remainder(NewProve).
proof(_,_,_):-
   write('No proof.'),nl,nl,fail.

% next_step(+Premise,+Consequent,+Terms,-Rule)
%   Constructs Rule which has Premise as antecedent and Terms as
%   Consequent, perserving the temporal orderings

next_step(Premise,Q,[],AssertRule):- !,
   sub_time(Premise,PremiseT),
   form_rule(PremiseT,[[Q]],AssertRule).
next_step(Premise,_,Consequent,AssertRule):-
   sub_time(Premise,PremiseT),
   add_time2(Consequent,ConsequentT),
   form_rule(PremiseT,ConsequentT,AssertRule).

form_rule(P,C,AR):-
   conj_to_list(Conj and nil,P),
   AR = Conj cause Conseq,
   member(Cons,C),
   member(Conseq,Cons).
```

```
% abduction(+Consequent,-Results)
%   Returns Results which is a list of terms which can conclude
%   Consequent by causal rules.

abduction(Consequent,Result):-
   exp_conseq(Consequent,Alts),
   add_time(Alts,AltsT),
   repeat_abduction(AltsT,Result).

repeat_abduction([],[]):- !.
repeat_abduction([H|T],[Alts|R]):-
   exp_conseq(H,Alts),!,
   repeat_abduction(T,R).

% exp_conseq(+Consequent,-Alternatives)
%    Given a Consequent, find a set, Alternatives, of events that
%    are the premises for the Consequent in existing rules

exp_conseq(Consequent,Alts):-
   Premises cause Consequent,!,
   conditions(Premises,Alts).
exp_conseq(nec(T,T,E),Alts):-
   Premises cause 'POTE#'(T,_,p-E),!,
   conditions(Premises,Alts).
exp_conseq(Consequent,[Consequent]).

conditions(P,[P]):-
   \+(P = _ and _),!.
conditions(P1 and P2,[P1|R]):-
   conditions(P2,R).

% exp_premise(+Premise,-ExpandPremise,-Prove)
%    Given Rule, finds ExpandPremise from existing rules that leads
%    to the same conclusion by proving the events in Prove set
%    E.g. Rule: a & b -> c and there exists a rule a & d -> e, then
%        make new premise e & b and prove d.

exp(P,EP,Complement):-
   conj_to_list(P,PList),
   exp_premise(P,EPP,C),
   diff_set(C,PList,Complement),
   remove_duplicates(EPP,EP).
exp(P,[P],[]).

exp_premise(P,EP,Complement):-
   exp_premise1(P,EP,[],Complement,[]).

exp_premise1(P,[Q|EPSofar],EPSofar,Comp,CompSofar):-
   \+(P = _ and _),
   Premise cause 'POTE#'(T+1,_,p-E),
   Q = nec(T,T,E),
   conjunct_member(P,Premise,C),
   conj_to_list(C,Complement),
   append(Complement,CompSofar,Comp).
exp_premise1(P,[Q|EPSofar],EPSofar,Comp,CompSofar):-
   \+(P = _ and _),
   Premise cause Q,
   \+(Q = 'POTE#'(_,_,_)),
   conjunct_member(P,Premise,C),
   conj_to_list(C,Complement),
   append(Complement,CompSofar,Comp).
exp_premise1(P1 and P2,EP,EPSofar,Comp,CompSofar):-
   Premise cause Q,
   conjunct_member(P1,Premise,C),
   conj_to_list(C,Complement),
   append(Complement,CompSofar,NewCompSofar),
   exp_premise1(P2,EP,[Q|EPSofar],Comp,NewCompSofar).

% conjunct_member(+Term1,+Conjunction,?Term2)
%   succeeds if Term2 is a conjunct of Conjunction (connected by 'and')
%   which has Term1 as its conjunct

conjunct_member(P,P and R,R):- !.
conjunct_member(P,H and P,H):- !.
conjunct_member(P,H and T,H and Complement):-
   conjunct_member(P,T,Complement).
```

```
conjunct_member(P,P,nil):- !.

% conj_to_list(+Conjunction,-List)
%   creates a List which consists of the conjuncts within Conjunction

conj_to_list(nil,[]):- !.
conj_to_list(A and B,[A|R]):- !,
   conj_to_list(B,R).
conj_to_list(A,[A]).

% valid_condition(+List,+EventList,-NewList,?Flag)
%   If Flag is set to 1, there is no valid event in the List;
%   NewList is a list of events in the List which is valid in the
%   context of events in the EventList

valid_condition([],_,[],F):- !,
   \+(F == 1).
valid_condition([H|T],EventList,NewList,F):-
   \+(F == 1),
   H = nec(t,t,E),
   member(E,EventList),!,
   valid_condition(T,EventList,NewList,F).
valid_condition([H|T],EventList,[H|NewList],F):-
   \+(F == 1),
   H = nec(t,t,E),
   valid_con(E,EventList),!,
   valid_condition(T,EventList,NewList,F).
valid_condition(_,_,_,1).

% valid_cause(+Rule)
%   succeeds when the causal relation within the Rule is valid

valid_cause(nec(_,_,E1) cause nec(_,_,E2)):-
   \+(E1 = E2),
   valid_c(E1),valid_e(E2).

%%  Utility predicates

% add_time(+List,-NewList)
%   NewList is a list of formulae in which all temporal terms the
%   formulae in the List are incremented by 1 time step

add_time([],[]):- !.
add_time([nec(t,t,E)|T],[nec(t+1,t+1,E)|R]):- !,
   add_time(T,R).
add_time([nec(t+S,t+S,E)|T],[nec(t+S1,t+S1,E)|R]):- !,
   S1 is S+1,
   add_time(T,R).
add_time(_,[]).

add_time2([],[]).
add_time2([H|T],[AD|R]):-
   add_time(H,AD),
   add_time2(T,R).

% sub_time(+List,-NewList)
%   NewList is a list of formulae in which all temporal terms the
%   formulae in the List are decremented by 1 time step

sub_time([],[]):- !.
sub_time([nec(t,t,E)|T],[nec(t,t,E)|R]) :- !,
   sub_time(T,R).
sub_time([nec(t+1,t+1,E)|T],[nec(t,t,E)|R]):- !,
   sub_time(T,R).
sub_time([nec(t+S,t+S,E)|T],[nec(t+S1,t+S1,E)|R]):-
   S =\= 1,!,
   S1 is S-1,
   sub_time(T,R).
sub_time(_,[]).

% Printing predicates

print_remainder([]):- !,
   nl.
print_remainder(P):-
   nl,
   write('And prove the following sentences:'),
```

```
    nl,
    printall(P).

printall([H|T]):-
    tab(3),
    write(H),
    nl,
    printall(T).
```