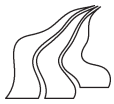
Influence Diagrams Involving Time,
a Framework for Decision Problems Involving Time

Mads Broe & Rune Jeppesen
broe@cs.auc.dk *flexi@cs.auc.dk*

Department of Computer Science
Aalborg University

10th June 2003

Aalborg University
Department of Computer Science, Fredrik Bajers Vej 7E, 9220 Aalborg Øst



TITLE:
Influence Diagrams Involving Time,
a Framework for Decision Problems Involving Time.

SUBJECT:
Decision Support Systems

PROJECT GROUP:
E2-101a

MEMBERS OF PROJECT GROUP:
Mads Broe
Rune Jeppesen

SUPERVISOR:
Thomas D. Nielsen

SEMESTER:
DAT6

PROJECT PERIOD:
1st of February 2003 to
10th of June 2003

NUMBER OF COPIES:
6

NUMBER OF PAGES:
105 pages
Total 109 pages

Abstract: In this Master thesis we study decision analysis, in which time on a quantitative level is an aspect. This study has, in addition to the studies of [Broe et al., 2003], resulted in a series of requirements for frameworks modelling decision problems involving time, DPITs. We present a framework for modelling and solving DPITs, which fulfills all of these requirements.
The framework, *influence diagrams involving time*, IDITs, was originally proposed in [Broe et al., 2003]. In this thesis it is extended to handle additional aspects of time, amongst these are local utility functions realized later than the end-time of the last decision. Furthermore, we devise a method for solving IDITs with respect to finding an optimal strategy. We solve an IDIT by resolving all asymmetries introduced by time. This leads to a number of symmetric sub-problems, which we solve using a method based on strong junction trees with lazy propagation.
We illustrate the solution method using two examples in order to demonstrate how the framework is used. Furthermore, we discuss the benefits and drawbacks of having one framework handling all issues of time as opposed to multiple smaller frameworks.

Preface

This Master thesis documents our project work on the Dat6 semester in decision support systems at the Department of Computer Science at Aalborg University. The project spans the spring semester of 2003, from the 1st of February 2003 to the 10th of June 2003. The project is a continuation of [Broe et al., 2003], which presents an analysis of the domain of decision problems involving time.

We would like to thank our supervisor, Thomas D. Nielsen, for outstanding supervision during the semester. We would also like to thank the other staff members in the Decision Support Systems group at Aalborg University for their willingness to answer any questions, we have had. Furthermore, we thank Søren Holbech Nielsen for enlightening discussions during the semester.

Mads Broe

Rune Jeppesen

Contents

Chapter 1

Introduction

Decision analysis is a research area focusing on how to take decisions, in uncertain surroundings, and do this in a manner, which is optimal for the decision taker. Trying to formalize decision analysis has resulted in a range of different frameworks each having their pros and cons.
[von Neumann and Morgenstern, 1944] and [Raiffa, 1968] proposed to structure a decision problem in a so-called *decision tree*. A decision tree models the choices and circumstances as internal nodes in a tree and the outcomes as the leaves. The tree gives a good intuition of a decision problem, but it is exponential in size, wherefore other more compact frameworks have been proposed. As opposed to decision trees, in which the states of each variable are nodes in the tree, the more compact frameworks groups these states in variables. [Howard and Matheson, 1981] proposed *influence diagrams*, which is, basically, a Bayesian network, as described in [Pearl, 1988], [Lauritzen, 1996], and [Jensen, 2001], augmented with decision and utility nodes. Other frameworks have been proposed as well, amongst these are *valuation networks*, described in [Shenoy, 1992]. Both influence diagrams and valuation networks are based on multiple levels, such that the reader can abstract from certain details on different levels of the framework.
Even though decision analysis has been researched for several decades, surprisingly little research has been done on the effects of time in decision problems. The three frameworks described above are all capable of representing a qualitative aspect of time, which specifies in which order decisions and observations are taken. However, the quantitative aspect of time has gotten little attention. The quantitative aspect of time is a representation of time, which is directly quantified in the decision problem, that is, the point in time when something happens can have an effect on what happens. For instance, time may influence the possible choices for a Friday night in town. If the football match begins at eight o'clock the choice of going to the stadium and seeing the game is not possible at ten o'clock. [Horvitz and Rutledge, 1991] and [Horvitz and Seiver, 1997] discussed how time may influence the utility functions of a decision problem, which is in

fact a quantitative aspect. In [Broe et al., 2003] the class of decision problems involving time was analyzed and a framework, for modelling these, was proposed. The analysis showed how decision problems involving time, combine two well studied aspects of decision problems, these being asymmetric problems and continuous variables. [Bielza and Shenoy, 1999] and [Nielsen and Jensen, 2002] both discuss the effect of modelling asymmetric decision problems. The effects of having continuous variables in Bayesian networks have gotten some attention, for instance, in the form of hybrid networks, [Lauritzen, 1996]. [Shachter and Kenley, 1989] and [Madsen and Jensen, 2003] examined continuous variables in influence diagrams, but, as for similar work, the continuous variables are restricted in different manners, for instance, by not allowing continuous variables as parents of discrete variables. [Lerner et al., 2001] showed one approach of removing this restriction for Bayesian networks.
One aspect of decision problems is the model, another is finding a strategy for taking the optimal choices of the decision problem. For each of the frameworks, described above, solution methods have been proposed. For solving influence diagrams [Howard and Matheson, 1981] proposed a method turning the influence diagram into a decision tree, which could be solved using the method from [von Neumann and Morgenstern, 1944]. [Shachter, 1986] and [Jensen et al., 1994] also proposed solution methods for influence diagrams. The solution method proposed by [Nielsen and Jensen, 2002] for asymmetric influence diagrams splits the asymmetric influence diagram in symmetric sub-problems, which are then solved using lazy propagation as described in [Madsen and Jensen, 1999]. A similar approach for splitting the problem is used in [Demirer and Shenoy, 2001] for a framework based on valuation networks and *sequential decision diagrams*. Furthermore, [Nielsen and Jensen, 2002] argues for the symmetric sub-problems being welldefined based on [Nielsen and Jensen, 1999].
The reason why time is an interesting aspect in relation to decision problems is that we aim at modelling the real world, and time is of importance, on some level, in almost everything we do. So being capable of modelling time as a factor of a decision problem yields models closer to reality. For example, modelling the decision of buying stocks should be done reflecting the time at which they are bought, as the point in time when they are bought influences the price, and later the earnings. Time not only influences the circumstances, but decisions and utilities as well. Chapter 3 gives an in-depth analysis of the influence of time.

1.1 Problem Specification

Representing and solving decision problems involving time is the main focus of this thesis. By further analysis of decision problems involving time we want to find important elements, which were not disclosed in [Broe et al., 2003]. The results of this analysis should end with an extended representation language for decision problems involving time, yielding it more suitable for modelling. Besides being more expres-

sive, we seek to devise a method for solving influence diagrams involving time of any given model of a decision problem involving time. Therefore, we describe a solution method which, given a model of a decision problem involving time, returns an optimal strategy for taking the decisions. Finally, we discuss the framework with respect to its usability.

1.2 Outline of the Thesis

This thesis consists of seven chapters, which can be divided into three main parts. Chapters 2 and 3 discuss the preliminaries of the framework being constructed. Chapter 2 gives an introduction to the notation used in this thesis, while specifying graph and probability related concepts. In it we also discuss how this thesis builds on [Broe et al., 2003], and where the main differences between the two works lie. Chapter 3 discusses the problem domain of decision problems involving time, introduces the concepts relating to these problems, and sets up a series of requirements for frameworks modelling decision problems involving time.
The second part of the thesis is focused on the development of the framework for modelling and solving decision problems involving time. Chapter 4 defines the representation language for modelling decision problems involving time, which is called influence diagrams involving time. The representation language is first described by its semantics, which is followed by definitions of its syntactical specifications. Finally, we argue for the representation language being welldefined. Chapter 5 describes a general solution method for solving influence diagrams involving time. The chapter gives an overview of the solution, and then specifies the details of each step involved in the method.
The final part concludes on the thesis by illustrating the use of influence diagrams involving time, and by discussing the pros and cons of a specific framework for modelling decision problems involving time. Chapter 6 introduces two examples, which are solved using the solution method proposed in Chapter 5. Furthermore, alternatives to the approximation method, we utilize, are discussed. In Chapter 7 we conclude on the thesis, by a discussion of the works of the thesis, and summarize the results achieved. Finally, we propose which aspects should be considered as future research.

Chapter 2

Preliminaries

In this chapter we describe the theory on which this thesis is built. The theory is divided into three sections, where Section 2.1 describes graph theory, and introduces the notation used in this thesis. Section 2.2 introduces the probability theory used and describes some of the theoretical aspects of utilities. Finally, Section 2.3 describes the representation language constructed in [Broe et al., 2003], and discusses how this thesis differs from the earlier work.

2.1 Graph Theory

As the modelling of decision problems is done using graphical models, graph theory is essential to any framework for modelling decision problems. We introduce the key aspects of graph theory used in this thesis.

An *element* is a generic term, which is used to cover any mathematical instance, for example, a variable or a graph. An unordered collection of distinct elements is referred to as a *set*. To separate a set from its elements we denote sets by bold capitalized letters, whereas elements are non bold, for instance, $\mathbf{X}$ could be the set consisting of X, Y and Z, also written as $\mathbf{X} = \{\mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$. In this thesis, unless explicitly stated, no sets are multisets, that is, sets do not include multiple instances of an element. We use traditional set operations when manipulating sets, such as $\cup$ for taking the union of two sets and $\cap$ to denote the intersection of two sets.

A *graph*, $\mathcal{G}$, is a pair of sets, $(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set of *nodes*, and $\mathbf{E}$ is a subset of $\mathbf{V} \times \mathbf{V}$, which we call *edges*. The edges of a graph can be *directed* or *undirected*. If both $(\mathsf{V}, \mathsf{V}')$ and $(\mathsf{V}', \mathsf{V})$, also denoted as $\{\mathsf{V}, \mathsf{V}'\}$, are in $\mathbf{E}$ the edge is undirected. If only one of these is in $\mathbf{E}$, it is a directed edge, which we usually refer to as an *arc*. If all edges of a graph are directed we call the graph a *directed graph*, and if not we call it an *undirected graph*.
Labelled graphs are graphs in which the edges are *labelled*. We define a labelled graph

by a triple, $(\mathbf{V}, \mathbf{L}, \mathbf{E})$, where $\mathbf{V}$ is a set of nodes, $\mathbf{L}$ a set of labels, and $\mathbf{E}$ a subset of $\mathbf{V} \times \mathbf{V} \times \mathbf{L}$.
A node, V, in a graph, $(\mathbf{V}, \mathbf{E})$, is said to be the *parent* of a node, V', if the arc (V, V') is in $\mathbf{E}$, and we say V' is a *child* of V. We denote the set of parents of some node, V, by $\mathbf{pa}(V)$, and the set of children of V as $\mathbf{ch}(V)$. If two nodes, V and V', are connected by an edge $\{V, V'\}$ they are referred to as *neighbours*. The set of neighbours of some node, V, is denoted $\mathbf{ne}(V)$.
In a graph, $(\mathbf{V}, \mathbf{E})$, a *path*, $\mathcal{P}$, is an ordered sequence of nodes, $V_1, V_2, \ldots, V_n$, in $\mathbf{V}$, where there exists an edge $\{V_i, V_{i+1}\}$ for $1 \leq i \leq n-1$. A directed path is a path consisting of only directed arcs. If there is a directed path, $V, V', \ldots, V''$, we say V'' is a *descendent* of V and V is an *ancestor* of V''. We denote the set of ancestors of some node, V, by $\mathbf{an}(V)$, and the set of descendents of V by $\mathbf{de}(V)$. Furthermore, a directed path, $V, V', \ldots, V''$, and an arc, (V'', V), is referred to as a *cycle*. If a graph does not have any cycles it is said to be *acyclic*.
A graph $(\mathbf{V}', \mathbf{E}')$ is a *subgraph* if there exists a graph $(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}'$ is a subset of $\mathbf{V}$, and (V, V') is in $\mathbf{E}'$, if and only if it is in $\mathbf{E}$ and both V and V' are in $\mathbf{V}'$. A graph, $(\mathbf{V}, \mathbf{E})$ is said to be *complete*, if, for all nodes, V, in $\mathbf{V}$, the set of neighbours equals $\mathbf{V} \setminus \{V\}$. A maximal complete subgraph is called a *clique*.

2.2 Probability Theory

In this section we introduce the probability theory used in this thesis. In the modelling of decision problems the choices are taken under uncertainty. This uncertainty is formalized using probability calculus.

2.2.1 Discrete Variables

A *discrete chance variable* is a finite set of mutually exclusive and exhaustive *states*, each of which is associated with a probability of being in that state. The semantics of chance variables in relation to decision problems are discussed in Chapter 3. Conventionally, we denote variables with capitalized letters, and its state by low case letters. For instance, V could be a variable with the states $\{v_1, v_2, \ldots, v_n\}$. We call the set of states of a variable the *state space* of the variable, which we denote $\mathbf{sp}(V)$. The uncertainty of a state of a chance variable, V, is represented by a *probability distribution* $P : \mathbf{sp}(V) \mapsto [0; 1]$, where it holds that:

$$\sum_{v \in \mathbf{sp}(V)} P(v) = 1.$$

We call a probability distribution over only one variable a *marginal probability distribution*.
The *joint probability distribution* of a set of chance variables, $\mathbf{V}$, is a function

$P_V : \mathbf{sp}(\mathbf{V}) \mapsto [0;1]$ for which it holds that:

$$\sum_{\overrightarrow{v} \in \mathbf{sp}(\mathbf{V})} P(\overrightarrow{v}) = 1,$$

where $\overrightarrow{v}$ is a configuration of the chance variables in $\mathbf{V}$. Given a joint probability distribution for a set of variables, $\mathbf{V}$, we can derive the joint probability distribution, P, for any subset, $\mathbf{V}'$, of $\mathbf{V}$ by *marginalizing* out the variables of $\mathbf{V} \setminus \mathbf{V}'$, that is:

$$P(\mathbf{V}') = \sum_{\mathbf{V} \setminus \mathbf{V}'} P(\mathbf{V}).$$

The state of a variable is always dependent on some context. For instance, the state of a variable, V, representing the second hand in a poker game is dependent on the first hand, represented by the variable, V'. The probability distribution of a chance variable, V, we say is conditionally dependent on V', written $P(V|V')$. Generally, a *conditional probability distribution* for some set of variables, $\mathbf{V}$, given a set of variables, $\mathbf{V}'$, is a probability distribution of $\mathbf{V}$ for each configuration $\overrightarrow{v'}$ of $\mathbf{V}'$. The conditional probability distribution $P(\mathbf{V}|\mathbf{V}')$ can by found using the *fundamental rule*:

$$P(\mathbf{V}|\mathbf{V}') = \frac{P(\mathbf{V}, \mathbf{V}')}{P(\mathbf{V}')}.$$

From the fundamental rule *Bayes rule* can be deduced:

$$P(\mathbf{V}|\mathbf{V}') = \frac{P(\mathbf{V}'|\mathbf{V}) \cdot P(\mathbf{V})}{P(\mathbf{V}')}$$

2.2.2 Continuous Variables

In the previous section we described chance variables for which the state space is finite, however, not all variables have a finite state space. *Continuous chance variables* are chance variables with an infinite state space. Unlike discrete chance variables, continuous chance variables do not have a probability associated with each state. Instead, we associate a *density function*, which reflects the probability distribution of the continuous variable. That is, the probability distribution of a continuous chance variable being in the interval $]a;b]$ is a function $f_V : \mathbb{R} \mapsto \mathbb{R}^+ \cup \{0\}$ for which it holds that:

$$\int_{-\infty}^{\infty} f_V(x)dx = 1,$$

and zero for all x not in $]a;b]$.
We define the probability of an interval in the continuous chance variable as a *cumulative probability distribution*, that is, the probability of a chance variable, V, being at most a is:

$$P_V(a) = \int_{-\infty}^{a} f_V(x)dx.$$

In this thesis all continuous chance variables are associated to a χ^2-distribution, which is given by:

$$f_V(x) = \begin{cases} \frac{e^{-\frac{x}{2}} \cdot x^{\frac{k}{2}-1}}{2^{\frac{k}{2}} \cdot \Gamma(\frac{k}{2})} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

where Γ is a gamma-distribution as described in [Grimmett and Stirzaker, 1992], and k is a measure of degrees of freedom. To illustrate the behaviour of a χ^2-distribution, Figure 2.1 shows it for one with three degrees of freedom.

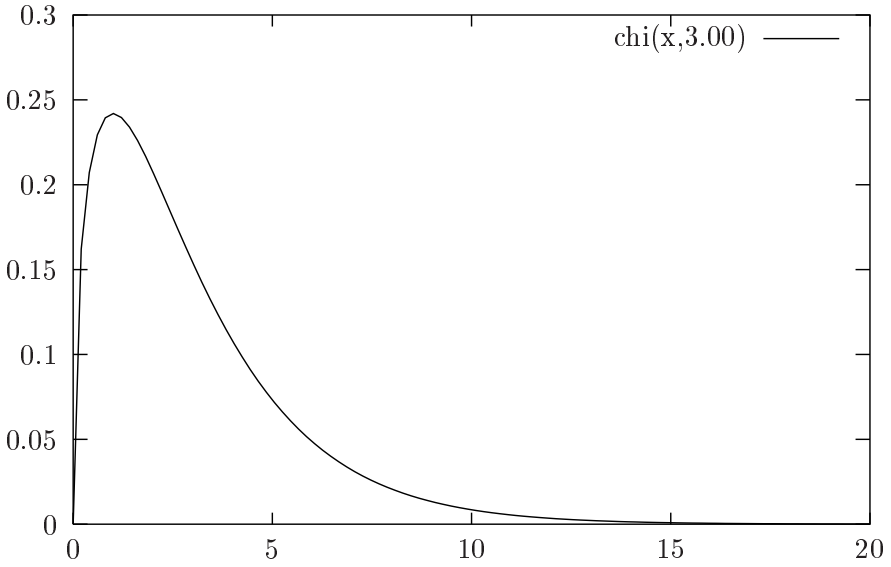


Figure 2.1: *A χ^2-distribution with three degrees of freedom.*

We later argue for the choice of using χ^2-distributions.

2.2.3 Utility Theory

In the domain of decision analysis, a value, or utility, denoted by u, is associated to each configuration of a set of variables, $\mathbf{V}$. This value reflects how preferable the configuration is in relation to other configurations over the set of variables. Some of these variables may be chance variables, that is, there is an uncertainty of the state of the variable. When the value is directly associated with a chance variable, it is possible to find the *expected value* or *expected utility* of this chance node given a configuration of the rest of the variables in the set.
The expected utility is the sum of the values weighed in accordance to the uncertainty

of the configuration, that is, the expected utility, EU, of a chance variable, V, with the states $v_1, v_2, \ldots, v_n$, which has a value $u_i(v_i, \overrightarrow{x})$ for $1 \leq i \leq n$, given a set of variables $\mathbf{X}$ which has a configuration $\overrightarrow{x}$ is:

$$EU(V) = \sum_{i=1}^{n} P(v_i) \cdot u_i(v_i, \overrightarrow{x}),$$

for all configurations of $\mathbf{X}$.

2.3 Previous Work

In this section we discuss the works of [Broe et al., 2003], which is the foundation for this thesis, and relate [Broe et al., 2003] to the work presented in this thesis.

[Broe et al., 2003] took outset in the identification of a class of decision problems, which involve time. The main difference in these problems as opposed to traditional decision problems is that time not only influences the structure of the decision problem, but also the numerical part. That is, where traditional decision problems have a qualitative aspect of time, influencing the order in which decisions are taken, decision problems involving time have a quantitative aspect too. Therefore, besides influencing the order, time can influence the uncertainty of a state of a chance variables or the value of a utility.
After an analysis of traditional frameworks for modelling decision problems, their capabilities of modelling decision problems involving time were analyzed. This resulted in a series of requirements for frameworks modelling decision problems involving time, and the conclusion that the traditional frameworks were not capable of modelling decision problems involving time in a satisfactory manner.
Instead, a new framework, influence diagrams involving time, abbreviated as IDITs, was constructed. IDITs are based on the framework of influence diagrams, which has been extended to handle the time issues required by decision problems involving time. The framework represented the quantitative aspect of time by introducing variables, representing time. Two variables were introduced, one modelling the possibility of controlling time, and one modelling the uncertainty involving time. Furthermore, influence diagrams involving time were made to represent time influencing utility functions and chance variables, and aspects of time involving the restrictions of decisions and the possibility of observations. [Broe et al., 2003] constructed the representation language of IDITs and proposed a sketch of how a possible solution could be found.

In this thesis we extend the representation language of [Broe et al., 2003], such that it handles some of the aspects of time, which were not included. And we clarify how the framework of IDITs actually handles time influencing utility functions and chance variables. The representation language of [Broe et al., 2003] was restricted in the sense that it does not allow time to influence the order in which decisions are taken,

but only to restrict the possible options of the decisions. Likewise the possibility of having time not associated to decision, is not included in [Broe et al., 2003]. It is however an interesting aspect of time as the payoff of a decision is sometimes postponed into the future, for instance, when selling stocks the actual payoff comes the next day. These clarifications and extensions yield a more expressive framework, which should be expressible through the syntax and semantics of the representation language.

Furthermore, we specify what a welldefined IDIT is, and how we ensure that this property is fulfilled. This property ensures that a unique decision can be identified as the next decision to be taken, and is, therefore, an essential matter when modelling decision problems involving time, and when solving them.

The solution sketch of [Broe et al., 2003] did not include a general description of how to solve decision problems involving time, and this matter is solved in this thesis. We describe how to solve an IDIT, and discuss the usefulness of the proposed solution method. Having specified a solution method makes it possible to implement and test the framework, which is a necessity if the framework should be usable, besides as a means of communication. Finally, we discuss how the representation language benefits as a means of communicating a decision problem involving time, and discuss the benefits and difficulties of using the framework.

Chapter 3

Decision Problems Involving Time

The purpose of this chapter is to recapture the ideas and parts of decision problems involving time, abbreviated as DPITs, as presented in [Broe et al., 2003]. DPITs are the foundation on which the rest of this thesis builds.

DPITs constitute a class of decision problems, in which time influences the decision taking. Unlike decision problems, time quantified and plays a central part of the uncertainty of observations, the order of events, and of preferences in DPITs. DPITs share the fact that the concept of time directly influences the parts of the problem, whether by increasing the uncertainty of an observation, yielding it impossible to make an observation, or waiting until a certain point in time before taking some decision. For instance, it may not be possible to observe the severity of an earthquake right away and by waiting a period of time before dispatching help, the help can be specialized and thereby save a lot of lives, however, the delay may cause many people to die.
The concept of time also introduces an uncertainty in itself. Actions, which may be executed with ease now, may be impossible to perform ten minutes from now, and unforeseen events may change the amount of time it takes to perform even a simple task.

In Section 3.1 we specify what a decision problem involving time, abbreviated as a DPIT, is based on and present the parts it consists of. In Section 3.2 we present the properties of DPITs, and in Section 3.3 we list requirements for a framework to model DPITs, and a justification for these.

3.1 Parts of Decision Problems Involving Time

A DPIT describes a collection of *circumstances* and *choices* and the association of these to the *decision taker*. It also encompasses information of the *uncertainty* of the circumstances given the choices, the *temporal order* of circumstances and choices, the *preferences* of the decision taker, and the time related to this information.
In the following sections we elaborate on these parts.

3.1.1 The Decision Taker

A decision taker is an entity, which encounters a series of choices, from which he chooses a subset, based on his preferences. He is always thought of as an entity, that is, if the decision taker represents a group of people, it is assumed that this group, with certainty, bases its choices on the same set of preferences.

3.1.2 Variables in a Decision Problem Involving Time

A decision problem consists of a set of variables, a utility function, and relationships between the variables. There are two types of variables, chance and decision variables. The set of chance variables is denoted as $\mathbf{V}_\mathsf{C}$ and the set of decision variables as $\mathbf{V}_\mathsf{D}$. A chance variable is comprised of a set of mutually exclusive and exhaustive circumstances, while a decision variable is comprised of a set of mutually exclusive and exhaustive choices. This in turn means that the circumstances of a chance variable and the choices of a decision variable each are conceptually related.
A choice is related to the decision taker as something over which he has direct control. A circumstance on the other hand is something over which the decision taker can only have an indirect control.

If a variable, V, is known to be in one of its states, ν, we call ν its *true state*, and we say that V is *instantiated* as being in ν.
We say that the decision taker can *choose* a choice or *take* a decision. Chance variables can be *observed* meaning that the decision taker knows their true state.

The set of chance variables for a DPIT consists of two disjoint subsets, one is a set of discrete chance variables, $\mathbf{V}_\mathsf{DC}$, and the other a set of time variables, $\mathbf{V}_\mathsf{T}$, that is, $\mathbf{V}_\mathsf{C} = \mathbf{V}_\mathsf{DC} \cup \mathbf{V}_\mathsf{T}$. The set of time variables, $\mathbf{V}_\mathsf{T}$, consists of two disjoint subsets, one being the set of end-times for decisions, denoted as $\mathbf{V}_\mathsf{T}^\mathsf{e}$, and the other the set of free time variables, denoted $\mathbf{V}_\mathsf{F}$, that is, $\mathbf{V}_\mathsf{T} = \mathbf{V}_\mathsf{T}^\mathsf{e} \cup \mathbf{V}_\mathsf{F}$. A time variable, T, in $\mathbf{V}_\mathsf{T}^\mathsf{e}$ is always *associated* with some decision variable, D, and, conventionally, we write the decision, to which the time variables is associated, as subscript, in this case $\mathsf{T}_\mathsf{D}^\mathsf{e}$. We say that $\mathsf{T}_\mathsf{D}^\mathsf{e}$ represents both the *end-time* of D and the *initiation-time* of the next decision to consider after taking D. The implications of this are discussed later. When it is apparent from the context what type of chance variable is referred to, we

use the term chance variable, otherwise we use their full name.
The set of decision variables, for a DPIT, also consists of two disjoint subsets, one being a set of discrete decision variables, $\mathbf{V}_{\mathrm{DD}}$, and the other a set of wait decision variables, $\mathbf{V}_W$. That is, the set of decision variables $\mathbf{V}_{\mathrm{D}}$ is defined as $\mathbf{V}_{\mathrm{D}} = \mathbf{V}_{\mathrm{DD}} \cup \mathbf{V}_W$. Wait decision variables are continuous decision variables. A wait decision is always a decision of how long to wait before taking the next decision, and we say that the wait decision is *referencing* a decision following it in the temporal order. When it is apparent from the context what type of decision variable we are talking about, we use the term decision variable, otherwise we use their full name.

Some choices may imply that a *timed action* is to be executed. The execution of a timed action takes some amount of time, thus time passes when executing the timed action. A decision variable consisting of one or more choices implying timed actions is called a *decision variable involving time*. A decision variable, which does not imply the execution of a timed action, is called an *instant decision*, and when a choice for an instant decision is executed, we call that choice an *action*. As actions do not impose the passing of time, the initiation-time and end-time of an instant decision is represented by the exact same point in time.
A decision variable involving time has a time variable associated with it, pinpointing how long the chosen timed action takes. As the choice indicates how long it takes to perform the timed action, the time variable represents an element of uncertainty in the execution of a timed action. The amount of time it is assumed to take for the timed action to be executed is known as the *time span* of the timed action. The actual time it takes is the difference in time between the initiation-time and the end-time of the decision at hand.
Furthermore, time variables represent the global time of the DPIT. That is, time accummulates through the DPIT for each observed time variable.

3.1.3 The Utility Function

Choices are taken on the basis of some set of preferences. These preferences can formally be described by a utility function. Before it makes sense to consider any DPIT the preferences of the decision taker should be clarified, and a method for doing this is to define a utility function for that DPIT. A utility function is a mapping of each configuration of the variables of the decision problem to a real number reflecting how preferable the configuration is. A utility function can be additively decomposed to a set of *local utility functions*. That is, a utility function, U, of some DPIT, can be defined as $\mathsf{U} = \sum_{i=1}^{n} \mathsf{u}_i$, where u_i is a local utility function. The set of local utility functions is denoted as $\mathbf{V}_{\mathsf{U}}$. Each local utility function maps the state spaces of a proper subset of the variables in the DPIT to $\mathbb{R}$, while U maps the state spaces of all variables in the DPIT to $\mathbb{R}$.

The specifications of a local utility function is, in principle, different for everybody, as it depends on the subjective preferences of the individual, that is, the preferences

of the decision taker. Therefore, a utility function can be any function, as long as it is unambiguous, and there is a surjection between the combinations of states of the variables influencing the utility function, and the number of possible outputs of the utility function.

3.1.4 Relationships Between Variables

The chance and decision variables of a DPIT are related through the uncertainty of the states of chance variables given decision variables. The joint conditional probability distribution for the chance variables in a DPIT given the decision variables is $\mathsf{P}(\boldsymbol{V}_\mathsf{C}|\boldsymbol{V}_\mathsf{D})$. The joint conditional probability destribution can be decomposed to a set of conditional probability distributions, one for each chance variable using the chain rule for influence diagrams.

Each choice is chosen based on a set of observations and the *relevant past* of that decision variable. The relevant past of some decision variable, D, is represented through the *observation function*, $\mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D})$. $\mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D})$ is the set consisting of the decision itself, the set of variables observed before D is taken at time t, and $\mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D}')$ where D' is the decision taken immediately before D.

Other relations between variables include how the effects of time may restrict the state space of a decision variable. This type of relationship is a restriction, which is represented by a restriction function. A restriction function on a decision, D, is a function, which maps the state spaces of a set of variables influencing D to some subset of the state space for D. That is, a function, $\gamma : \mathbf{sp}(\boldsymbol{V}) \mapsto \mathbf{sp}(\mathsf{D})$, where $\boldsymbol{V}$ is the set of variables restricting D. The set of restriction functions for a DPIT is denoted as $\boldsymbol{Q}_\mathsf{t}$. The subtyped t denotes that $\boldsymbol{Q}_\mathsf{t}$ is defined for every point in time, t.

A restriction function between variables can also refer to a chance variable only being observable within a specific time frame. For instance, if some test is performed and the result of the test is available to the decision taker after ten hours. The decision taker cannot use the information given by the best result, if he takes a decision after only five hours of the test has been performed. The test result is still available after ten hours, but it simply has no influence on the decision it was meant for, as the decision has already been taken.

We constrain decision variables to include at least one choice for any given point in time. This choice should still be conceptually related to the other choice, in such a manner that the extra choice makes sense. In other words, the decision taker must always be able to choose at least one choice for every decision variable.

3.2 Properties of Decision Problems Involving Time

DPITs have some properties as a consequence of time in these problems. This section describes these properties.

3.2.1 The No-Delay Assumption

A time variable associated with a decision represents both the end-time of this decision variable and the initiation-time of the next decision variable, we call this the *no-delay* assumption. The no-delay assumption states that between two decisions time is fixed. This assumption ensures that observing chance variables is instantaneous, that is, chance variables are only observed immediately before a decision variable at the point in time specified as the initiation time for that decision.

3.2.2 Temporal Order of Variables

The order in which decisions are taken in a DPIT, constitutes a temporal ordering of these and all other variables. For instance, a decision on whether or not to harvest crops on a field should not precede the decision of whether or not to sow the crops. Not all cases of the orderings of decisions are as apparent as just illustrated though, therefore an ordering of the decisions is specified for a DPIT. This ordering orders the decisions and time variables, such that, the choice of one decision or possibly the end-time of this decision makes it possible to unambiguously identify which decision is to be taken next. DPITs allow two or more decisions to be unordered initially with respect to each other, if, before taking any of the unordered decisions, a unique order can be found.
Furthermore, decisions are ordered in relation to the time variables, such that, the end-time of a decision is only known *after* the decision. We extend this to say that the end-time of a decision is known *before* taking the next decision in the DPIT. Observed chance variables, which influence some decision, D, are placed immediately before D in the temporal ordering, unless they have already been placed somewhere, that is, they influence another decision, which is placed before D.
DPITs do not have a total ordering of variables. However, as observations and decisions are taken, an ordering emerges. Instead of the total ordering there is a partial ordering, denoted as $\rightarrow$, of decision and chance variables. This ordering orders the decisions and time variables in relation to each other, and the discrete chance variables accordingly, but the discrete chance variables are not ordered with respect to themselves.
When a variable, V, is said to be *before* another variable, V', in the temporal order of a DPIT, V is either observed or taken before V', depending on whether V is a chance or decision variable. And if V is said to be *after* V', V is observed or taken after V'. When we write $\mathsf{V} \rightarrow \mathsf{V}'$, it means that V is before V' in the temporal order

of the DPIT. If we want to be more specific and express that V is immediately before V', we say so explicitly. For any two variables, V_1 and V_2, in $\mathbf{V}_\mathsf{D} \cup \mathbf{V}_\mathsf{T}^\mathsf{e}$, there exists an ordering, this is a transitive ordering, that is, if $V_1 \to W$ and $W \to V_2$, where W is some other variable in $\mathbf{V}_\mathsf{D} \cup \mathbf{V}_\mathsf{T}^\mathsf{e}$, then it follows that $V_1 \to V_2$.
Furthermore, there exists a total ordering of all time variables, as a consequence of time variables representing a global time aspect. This means that for every two time variables, T and T', in $\mathbf{V}_\mathsf{T}$ are ordered such that either $\mathsf{T} \to \mathsf{T}'$ or $\mathsf{T}' \to \mathsf{T}$.
Furthermore, if a time variable is a free time varaible, all discrete chance variables influence this time variable are said to be *prior* to the time variable and are therefore before the time variable in the temporal order.

The order of decision variables in a DPIT can be defined through the $\mathbf{obs}_{\overrightarrow{\mathsf{t}}}$ function for all decision variables in a DPIT, so some decision variable, D, is before some other decision variable, D', iff $\mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D}) \subseteq \mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D}')$. Through this ordering the first decision variable of a DPIT can be found, and we define the initiation-time of this decision variable to be zero.

- There is one decision, D, and a relation, $\mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D}) \subseteq \mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D}')$, for all $\mathsf{D}' \in \mathbf{V}_\mathsf{D} \setminus \{\mathsf{D}\}$. We refer to D as the *first decision variable* of the DPIT. Furthermore, there are no time variable influencing D, yielding the initiation-time of D as zero.

Let D and D' be decision variables, such that $\mathsf{D} \to \mathsf{D}'$, and let $\mathsf{T}_\mathsf{D}^\mathsf{e}$ and $\mathsf{T}_{\mathsf{D}'}^\mathsf{e}$ be the time variables associated with D and D', respectively. Then the following bullets comprise what can be deduced from having the temporal order of chance and decision variables in a DPIT.

$\mathsf{T}_\mathsf{D}^\mathsf{e}$ represents the end-time of D, and D is immediately before $\mathsf{T}_\mathsf{D}^\mathsf{e}$ in the temporal order.

- There is no variable, V, in $\mathbf{V}_\mathsf{D} \cup \mathbf{V}_\mathsf{T}$, such that $\mathsf{D} \to V \to \mathsf{T}_\mathsf{D}^\mathsf{e}$.

The end-time of D is less than or equal to the initiation-time of D'. If D is immediately before D', then, because of the no-delay assumption, the end-time of D is equal to the initiation-time of D'. That is, the timed action imposed by taking D must end before D' is initiated. And, consequently, $\mathsf{T}_{\mathsf{D}'}^\mathsf{e}$ is greater than or equal to $\mathsf{T}_\mathsf{D}^\mathsf{e}$.

- For all t_D, $\mathsf{t}_{\mathsf{D}'}$, and $\mathsf{t}_{\mathsf{D}''}$ in $\mathbb{R}$, where $\mathsf{T}_\mathsf{D}^\mathsf{e} = \mathsf{t}_\mathsf{D}$, the initiation-time of D' is $\mathsf{t}_{\mathsf{D}'}$, and $\mathsf{T}_{\mathsf{D}'}^\mathsf{e} = \mathsf{t}_{\mathsf{D}''}$, it follows that, $\mathsf{t}_\mathsf{D} \leq \mathsf{t}_{\mathsf{D}'} \leq \mathsf{t}_{\mathsf{D}''}$. Furthermore, if D is immediately before D', then $\mathsf{t}_\mathsf{D} = \mathsf{t}_{\mathsf{D}'}$.

If D' is a wait decision variable, then there is always a time variable, $\mathsf{T}_{\mathsf{D}'}^\mathsf{e}$. If D is immediately before D', then $\mathsf{T}_{\mathsf{D}'}^\mathsf{e}$ always represents a point in time, which is later

than or equal to T^e_D plus the amount of time the decision taker has chosen to wait in D', this we refer to as the *delay* of D'. In short the point in time D' ends is always the same as or later than the initiation-time in addition to the delay period chosen at D.

- If there exists a wait decision variable, D', then $D' \rightarrow T^e_{D'}$ and, for all t_i, d, and t_D in $\mathbb{R}$, where the initiation-time of D' is t_i, $D' = d$, and $T^e_{D'} = t_D$, it follows that $t_i + d \leq t_D$.

When T^e_D, is immediately before D' in the temporal order, T^e_D represents the initiation-time of D'. This also holds if there are a number of decision variables and no other time variable between T^e_D and D', T^e_D then represents the initiation-time and end-time of all intermediate decision variables, and these intermediate decision variables represent instant decisions. That is, the point in time represented by T^e_D also represents the initiation-time of D'.

- If $T^e_D \rightarrow D'$ and there is no time variable, T', such that $T^e_D \rightarrow T' \rightarrow D'$, then, for all t and t_D in $\mathbb{R}$, where $T^e_D = t$ and the initiation-time of D' is t_D, it follows that $t = t_D$.

A result of the no-delay assumption and the fact that free time variables exist is that a time variable, T, does not have to be influenced by the choice chosen at some decision variable, D, but if this is the case, then no new decision variables may be considered in the DPIT, as the no-delay assumption would be violated.

- If, for two time variables, T_1 and T_2, where $T_1 \rightarrow T_2$, there is no decision variable, D, such that T_1 is associated with D, then there are no decision variable D', such that $T_2 \rightarrow D'$, and T_1 and T_2 are considered free time variables.

This rule allows for several time variables to influence each other while not being influenced by or influencing any decision variables. This means that utility functions may be influenced by time, but not by the decision taker himself. This represents a phenomenon we call a *post-realized utility function*.

As mentioned, the temporal order of a DPIT includes not only ordering time variables and decisions, it also includes the order of observing chance variables, which constitutes a partial temporal order. Meaning that there is no predefined order of observing chance variables, when taking decisions. The only rule is that, according to the no-delay assumption, the observation of chance variables is instantaneous.
Chance variables, which are only observable in some specific time interval, have a special role in the temporal order. Such chance variables can be observed only within this specific time span.

3.2.3 Decision Scenarios

A *decision scenario* for a DPIT is a list of circumstances and choices, which has a utility attached to it. Each circumstance and choice in a decision scenario represents a state of one variable in the DPIT, and a decision scenario respects the temporal order of the DPIT. For each variable, represented in a decision scenario, we work under the assumption that the decision taker has information of the past and future of that variable. The future of a decision variable, D, is all decisions, which are to be taken after D.
A decision scenario is a *configuration* of a subset of variables in a DPIT. The maximum number of decision scenarios of a DPIT equals the Cartesian product of the state space of all variables. Therefore, in any DPIT there are an infinite number of configurations, because of the continuous variables.

3.3 Representing Decision Problems Involving Time

As a consequence of introducing time, DPITs cannot be modelled by traditional frameworks for modelling decision problems, as they do not present these models in a compact nor a complete manner. [Broe et al., 2003] showed that traditional frameworks tend to clutter with arcs when a DPIT is attempted at being modelled, and the resulting models do not correctly model the continuity of time. Furthermore, DPITs are asymmetric, as a consequence of restrictions of decisions at given points in time, which needs special frameworks, such as those presented in [Bielza and Shenoy, 1999] and [Nielsen and Jensen, 2002], while still representing time correctly.
There is a possibility that some, or all, variables may be influenced by time, or influence time themselves. In order to use a framework as a means of modelling DPITs, these aspects must be expressible in the framework. Therefore a series of requirements was proposed in [Broe et al., 2003], which, when respected, handles these aspects.

The requirements proposed below are presented as rules for constructing a framework for representing and communicating DPITs.

3.3.1 Requirements for Frameworks

The requirements presented here originate from [Broe et al., 2003], in which they were concocted through an analysis of frameworks. This analysis resulted in the discovery of problems with traditional frameworks when attempting to use them for modelling a DPIT. The requirements only reflect what should be expressible to model the aspect of time, but a framework for DPITs should be capable of modelling any decision problem, that is, it should not lose any expressive power in the effort of modelling DPITs.

This description of the requirements is presented by first showing that there in fact is a problem, then the requirement for solving the problem is presented, and finally the requirement is explained more thoroughly.

When considering DPITs it is apparent that a means of representing the passing of time, and time itself, is needed. There should be both an element of time, which is controllable by the decision taker, and one, which has an element of uncertainty to it, in order to handle, for instance, unforeseen delays in performing a timed action. This need leads to Requirement 1.

Requirement 1
It should be possible to model time and wait decision variables. That is, the variables in $\mathbf{V}_W \cup \mathbf{V}_\mathsf{T}$*, should be expressible in the framework.*

The introduction of these types of variables introduces the risk of having a framework, which is hard to interpret, as additional types of variables have to be represented. This is acceptable though, as this requirement yields a framework capable of representing time explicitly. This requirement also makes certain that time is represented by continuous variables, as variables in $\mathbf{V}_W \cup \mathbf{V}_\mathsf{T}$ are continuous.

As time often has an effect on what choices a person is presented with, a framework for modelling DPITs has to be capable of representing decision variables, for which the state space varies according to the point in time at which they are taken. Requirement 2 introduces such possibilities.

Requirement 2
It should be possible to model decision variables, for which the state space varies over other variables, and accurately portray the dependencies involved. That is, the domains of restriction functions in $\mathbf{Q}_\mathsf{t}$ *should be expressible in the framework.*

A restriction function is a function, $\mathsf{Q}_\mathsf{t} : \mathbf{sp}(\mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{D}) \setminus \{\mathsf{D}\}) \times \mathbb{R} \mapsto \mathbf{sp}(\mathsf{D})$, where D is the decision variable, which is being restricted, and $\mathbb{R}$ represents time.

As described above, time variables are continuous chance variables. The time span of a timed action can be affected by other circumstances, for instance, the weather can have an impact on how long harvesting some field takes. This calls for the need to represent variables, which alter the end-time of decisions.

Requirement 3
It should be possible to model variables affecting the end-time of a decision. That is, for every variable, T*, in* $\mathbf{V}^e_\mathsf{T}$*, the domain of the density function for* T*, obtained from* $\mathsf{P}(\mathbf{V}_\mathsf{C}|\mathbf{V}_\mathsf{D})$*, should be expressible in the framework.*

This requirement states that the state space of a time variable may be restricted and this should be expressible in the framework. Therefore, time variables can be conditionally dependent on other variables, shown at least through the density function of the time variable.

In order to keep the model of a DPIT unambiguous, the next decision to be taken and the observations for this decision must be identifiable. Furthermore, it should be possible to always know what the decision taker bases his choices on, that is, for different points in time different variables may be observed.

Requirement 4
It should be possible to model the time dependent observation function. That is, for all decisions, D, *in* $\mathbf{V}_D$, $\mathbf{obs}_{\vec{t}}(D)$ *should be expressible in the framework.*

This requirement ensures that for all decisions, D in $\mathbf{V}_D$, and all points in time, t, there exists a function, $\mathbf{obs}_{\vec{t}}(D)$, giving the set of variables, which are observed for any D at the point in time t.

Having time influence discrete chance variables is an aspect, which the framework should be capable of modelling. That is, to have the possibility of having probability distributions change over time.

Requirement 5
It should be possible to model time variables having an impact on discrete chance variables. That is, the existence of a conditional probability distribution, $P(C|\mathbf{X})$, *for some chance variable,* C, *in* $\mathbf{V}_{DC}$, *for which some time variable,* T, *is in the conditioning set,* $\mathbf{X}$, *should be expressible in the framework.*

This requirement introduces the concept of having discrete chance variables have a time variable in their conditioning set. This way time not only influences what the decision taker can control, but also the circumstances of the DPIT, over which he has no direct control.

It is often not easy to comprise the preferences of a decision taker into one meaningful expression, which can be calculated through a single utility function. By introducing the concept of local utility functions, the preferences can be represented in a manner, which is more easily understood. Different points in time can also influence the preferences of the decision taker, this should also be expressible in the local utility functions, thereby giving the framework more expressive power.

Requirement 6
It should be possible to model variables determining the value of local utility functions. That is, the domain of all local utilities, u_i, *obtained from* $\mathbf{V}_U$, *where* $i = \{1, 2, \ldots, n\}$, *should be expressible in the framework.*

The local utility functions should be expressible, and should each encompass different parts of the whole DPIT. This yields models more easily read and interpreted, and gives a set of local utility functions, which can be realized at different points in time.

In addition to these six requirements for modelling time, [Broe et al., 2003] presented three requirements, which were the main requirements focusing on the presentability

of the models modelled using the framework. These are: unambiguity, compactness, and easily read by humans. These requirements impose guidelines for frameworks, representing DPITs. The representation language should present DPITs in a manner, which does not confuse the reader by having redundant elements, which can be misinterpreted. The representation language should consist solely of the elements necessary for giving the decision taker the correct interpretation of the DPIT at hand and the representation language should be presented in a manner, which is intuitive for a human when examining the model.

We have, furthermore, found two requirements, which ensure that a framework for modelling DPITs also can model post-realized utility functions and decisions, for which the order of taking them changes due to the point in time they are to be taken. In many real world decision problems the payoff of taking a tacision is not necessarily realized right after taking the last decision of the decision problem. For example, the total cost of a loan is not cecessarily known when taking the loan, as the iterest rates fluctuates. Requirement 7 ensures this is modellable.

Requirement 7
It should be possible to model time variables, which are not associated to decision variables, but have an effect on some local utility function. That is, having a time variable, T*, in the domain of some local utility function,* U*, should be expressible in the framework.*

As it is not always the case that decisions are taken in the same order, a framework for modelling DPITs should make it possible to have time influence the order of decisions.

Requirement 8
It should be possible to have time variables affect the temporal ordering of two decisions. That is, $\mathsf{D} \rightarrow \mathsf{D}'$ *at some points in time and* $\mathsf{D}' \rightarrow \mathsf{D}$ *at all other points in time, should be expressible in the framework.*

As we further require that we at any point in time know what decision is next, this should also be ensured in the framework.

3.4 Summary

Through this chapter we have defined the concept of a DPIT. We have done this by introducing the parts of a DPIT, and discussing which properties DPITs have. Finally, we have presented guidelines for how an every day problem can be expressed in terms of a DPIT, and we have set up requirements, which help formulate a framework for representing DPITs.

Chapter 4

Representing Influence Diagrams Involving Time

In Chapter 3 a special class of decision problems, which cannot be modelled using the frameworks normally used for modelling decision problems was introduced. Modelling DPITs requires that the representation language can model both the asymmetries and the continuous elements of DPITs. In [Broe et al., 2003], a framework, IDITs, which was tailored to represent these, was presented. In this chapter we recapture the findings of [Broe et al., 2003] and extend the existing representation language to handle the additional requirements.
We give an informal description of IDITs in Section 4.1, in which we present both the qualitative and the quantitative levels in an informal manner and present examples for clarification. In Section 4.2 we present the formal description of the aforementioned levels, and present definitions of the essential elements in this representation language.

4.1 Informal Description of Influence Diagrams Involving Time

In this section we give an informal introduction to IDITs, which is the framework for representing DPITs. We only discuss the representation language of the framework and postpone the solution method to Chapter 5. We give semantics whenever elements are introduced, and discuss how a DPIT is modelled using an IDIT. When referencing elements directly related to DPITs, we do not specify the semantics again, instead we refer the reader to Chapter 3 for these.

4.1.1 Description of the Parts of Influence Diagrams Involving Time

IDITs were introduced to represent DPITs in a manner, which is compact, unambiguous, and easy to read for humans.
The framework is, as the name implies, based on influence diagrams [Howard and Matheson, 1981] and uses much of the same terminology. The representation language is divided into a *qualitative* and a *quantitative part*. The qualitative part is a directed labelled graph describing global information regarding relations between variables and utility functions, and the quantitative part describes local information relating to each variable or local utility function. We describe each part in turn beginning with the qualitative part.

The qualitative part of an IDIT is a directed labelled graph consisting of nodes, representing variables and local utility functions, and arcs, representing relationships between these. The qualitative level of IDITs gives the reader an overview of the DPIT without including numerical information for the variables, yielding it easy to communicate. The nodes in an IDIT are divided into five sets in accordance to the type of variable or function they represent. The five sets of nodes are: *chance nodes*, *time nodes*, *decision nodes*, *wait decision nodes*, and *utility nodes*.
A decision node represents a decision variable from the DPIT. Graphically a decision node is drawn as a rectangle. It can have a time node attached to it, representing that it is a decision variable involving time, or not, if it represents an instant decision.
A wait decision node represents a wait decision. A wait decision node is drawn as a double rectangle with a double semicircle attached. A wait decision node always has a double semicircle attached to it and always has the decision it is referencing as a direct child of the attached time node. Sometimes we refer to decision and wait decision nodes simply as decisions.
A chance node represents a chance variable, and is illustrated by a circular node. If a chance node represents a chance variable dependent on time, there is an arc from a time node to the chance node.
A time node represents either the end-time of the decision it is attached to, or the point in time some utility function is realized. A time node, representing the end-time of a decision, is represented, graphically, in an IDIT by a double semicircle attached to a decision node. A free time variable is represented by a double circle.
Local utility functions are represented in IDITs by utility nodes. A utility node is drawn as a diamond shaped node. If a local utility function is dependent on time there is an arc from a time node to the utility node.
Conventionally, we use a two letter abbreviation of the variable or function name as a unique identifier of a node. When discussing IDITs we usually do not distinguish between a node and the variable it represents, but if a distinction is needed, we refer to the node by the abbreviation.

The nodes in an IDIT are connected by arcs, which, depending on the node the arc emanates from, or is going into, have different semantics. We distinguish between five categories of arcs, which are: *informational arcs*, *dependence arcs*, *functional*

arcs, *guarded arcs*, and *restriction arcs*.
Informational arcs are arcs going into a decision or wait decision node and are drawn as solid arcs. An informational arc represents two related concepts. It represents a precedence of the nodes it connects, that is, the node from which an informational arc emanates precedes the decision to which it goes in the temporal ordering. It also represents that the decision taker has knowledge about the variable, from which the arc emanates, before taking the decision. Having knowledge about a variable, means to either have observed it as being in a specific state, if it is a chance or time variable, or to have decided upon it when it is a decision variable. Like in influence diagrams, IDITs operate under the *no-forgetting assumption*, [Howard and Matheson, 1981]. The no-forgetting assumption specifies that the true states of variables taken or observed before taking the current decision are remembered, such that arcs from those variables are omitted.
An arc going into a chance node, indicates a probabilistic dependence between the chance node and the node, from which the arc emanates. We call these arcs dependence arcs. The chance variable, the arc goes to, is conditionally dependent on the variable, from which the arc emanates. The absence of a dependence arc indicates that the chance variable is conditionally independent of the variable given its parents.
A functional arc is an arc going to a utility node. A functional arc specifies that the local utility function has the variable, from which the arc emanates, as one of its arguments. If the node is a time node, the arc specifies that the utility node represents a time dependent utility node.
A guarded arc is an informational arc associated with a boolean function. A guarded arc represents that the node, the arc emanates from, is only observed or decided upon in the time spans satisfying the function. The boolean function is referred to as a *guard*, and the guarded arc is drawn as a labelled arc in the IDIT. Guards are restricted to those involving time, meaning that a guard must reference a time variable in order to be evaluated. Instead of explicitly stating which time variable is referenced, it is by definition given as the time variable representing the initiation time of the decision to which the guarded arc goes. As long as the guard on an arc is satisfied the arc has the same semantics as an informational arc. We do not allow guards on dependence arcs.
A restriction arc indicates that the true state of the variable, the arc emanates from, restricts the state space of the variable, the arc is going into. Restriction arcs can only go to decisions, as restrictions on chance variables are emulated by setting the probability of the illegal states to zero. A restriction arc represents both an informational arc and the restriction of the decision, the arc goes to. Restriction arcs are drawn as dashed arcs.

For IDITs we assume that chance and decision nodes may not be barren nodes. That is, all chance and decision nodes have at least one child. [Shachter, 1986] argues that the removal of barren nodes is permitted. Furthermore, as it semantically does not make sense to have one variable or utility function being realized at two

different points in time, IDITs do not allow a node to be the child of more than one time node. Finally, we do not allow utility nodes as parents of other nodes. The past of a node, V, is the set of nodes which are before V in the temporal ordering. For DPITs we specified the set of observed variables, $\mathbf{obs}_{\overrightarrow{t}}(D)$, for some decision, D, and the set of prior variables, $\mathbf{prior}(T)$, for some free time variable, T. For IDITs the sets are defined as follows: $\mathbf{obs}_{\overrightarrow{t}}(D) = \{V|(V,D) \in \mathbf{E}, V \in \mathbf{V_C}\}$; $\mathbf{prior}(T) = \{V|(V,T) \in \mathbf{E}, V \notin \mathbf{obs}_{\overrightarrow{t}}(D)$ for any $D \in \mathbf{V_D}, V \in \mathbf{V_C}$ and $T \in \mathbf{V_T}\}$.
That is, the set of variables in $\mathbf{obs}_{\overrightarrow{t}}(D)$, is the set of chance variables, which have an informational arc going to D. If there is a guard on the informational arc, the variable, it emanates from, is in $\mathbf{obs}_{\overrightarrow{t}}(D)$ only if the guard is evaluated to true, given the configuration of the last observed time variable. The set of variables in $\mathbf{prior}(T)$ is the set of variables, which have a dependence arc going to T, and which are not in the set of observed variables for any decision.

As opposed to influence diagrams IDITs are allowed to include cycles, if guards ensure the cycles are broken, given any configuration of the past of the node the guarded arc goes to. Thus, when solving the IDIT, it results in an acyclic graph. That is, if a cycle exists there needs to be two guards in the cycle, which are mutually exclusive. The arcs in IDITs further constitute the partial ordering of all nodes ensuring there is a path containing all time and decision nodes. If a cycle between two decision nodes exists, the order of these can only be deduced when a configuration of the past is given.

In order to clarify the graphical representation of an IDIT, we have chosen to redefine the semantics of guarded arcs, from the definition presented in [Broe et al., 2003]. In [Broe et al., 2003], guarded arcs were inherited throughout the IDIT until the guard was satisfied, or a new arc was introduced, such as the arc, $(C, D_n, \mathtt{true})$, in the graph of Figure 4.1. This definition is, however, not intuitive as the guard, g, may never be true as a consequence of time only progressing. For instance, if g is $t \leq 4$, and the point in time at D_1 is taken is five, the guard would never become true for any of the following decisions.
Instead, we require there being an explicit arc in the IDIT if it should be observed at a later point, in the case where a guard on an arc has been evaluated to $\mathtt{false}$.

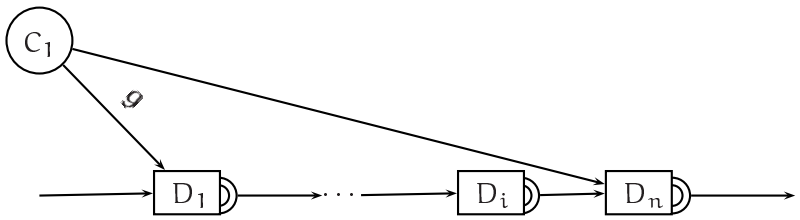


Figure 4.1: *A guarded arc,* g*, from a chance node,* C_1*, which is rendered obsolete as a new arc is introduced from* C_1*.*

The model of Figure 4.1 is still legal, but it means something different. Now, even if g is satisfied somewhere in between D_1 and D_n, C_1 is not observed when taking D_i.

To illustrate the elements of an IDIT, we present Example 1, in which we discuss a DPIT we call the *Search and Rescue Problem*.

Example 1
In this example we describe a DPIT, which revolves around the search and rescue (SAR) mission, taking place whenever a person is reported missing in a specific area. In this example, this area is known as *Lost Dale*. The decision taker of this problem is the SAR director of Lost Dale.
The area around Lost Dale is a mixture of forest and mountains, where people occasionally lose themselves in the valley. Whenever a person is reported missing, the SAR director assesses the situation at hand. Based on, amongst other things, the person missing, rescue teams are dispatched to find the missing person. After some time it is possible to get a heat signature of the entire area, giving an indication of where the missing person is. If a missing person is found, the SAR director receives a reward based on the success of the mission and the condition of the person, that is, if the person is alive or dead.

The SAR director has three decisions in this decision problem, namely *Mobilization (Mo)*, *When to begin searching (Ws)*, and *Search (Se)*. As many people visit Lost Dale and it often happens that somebody gets lost for a couple of hours, a SAR mission is first initiated 12 hours after the actual report of the missing person has been filed.
Lost Dale does not itself have SAR teams, but can issue some from neighbouring towns. The assembly of SAR teams therefore takes time, and the SAR director therefore has the possibility of mobilizing the teams before the initial 12 hours have passed.
Mobilization is a decision of which teams, if any, should be mobilized when a person is reported missing. The possible choices of the decision are *none*, *SAR dogs*, *helicopters*, and *both*. *both* being the combination of sending for helicopters and dogs. SAR dogs are capable of searching the forest, whereas a helicopter cannot see through the thick foliage, but it is better at searching the mountains. The choice of *Mobilization* influences when the search can begin as it takes time to assemble the teams. It takes at least 12 hours to get a helicopter to Lost Dale, and 18 hours to get dogs. When both SAR dogs and helicopters are needed two additional hours are used to get a joint strategy, thus the assembly of *both* takes 20 hours.
When to begin searching is a wait decision, which postpones the actual search decision until the SAR teams are assembled. As it is a wait decision it has a continuous state space.
Search is the decision in which the SAR director chooses in what part of Lost Dale to concentrate the search. As Lost Dale is part forest and part mountains the state space of the decision is *nowhere*, *forest*, *mountains*, and *both*, where *both* is a combination of searching both the forest and the mountains. *nowhere*, is the only choice available until the point in time the teams have been mobilized. Searching through the forest using the dogs takes 36 hours, whereas it is not possible to do a search of the forest using only helicopters. Searching the mountains with helicopters takes 18 hours, while it takes 80 hours searching the mountains using only the dogs.

At the choice of which team to mobilize the profile of the person missing is observed. The SAR director classifies missing persons into three categories, which are modelled in a chance variable, *Missing person (Mp)*, these categories are: the *lost girl* category, the *average male* category, and the *eager danger seeking male* category. These are also the names of the states of the variable. *Missing person* influences a chance variable, *Location of the missing person (Lp)*.

Location of the missing person is the actual location of the person. The states of the chance variable are *forest*, and *mountains*. This chance variable influences three other chance variables, namely *Survivability (Su)*, *Found (Fo)*, and *Heat signature (Hs)*.
Survivability represents the chance of the missing person being alive or dead when found. This chance variable is also influenced by the weather and the amount of time the search takes. The states of the chance variable are *alive*, and *dead*. The reward for the SAR director is dependent on the state of this variable.
How well the search has gone is modelled in *Found*, that is, if the missing person is found or remains lost. Besides *Location of the missing person*, the chosen area for the search, and the time the search ends, influences this chance variable. The two states of this variable are *found* and *lost*. The reward for the SAR director is also dependent on this variable.
The possible heat signature is modelled through the chance variable *Heat signature*. If the SAR director waits 48 hours before taking the decision on where to search, he has a heat signature of Lost Dale, indicating the location of the missing person.
In Lost Dale the Sun normally shines at least six days of the week, and as it has never rained or snowed two days in a row, it is assumed that it rains or snows at most one day each week. *Weather (We)* is the chance variable, which models if it will rain or snow one day during the search, or if it will stay sunny. It has three states: *sunshine*, *rain*, and *snow*. If it rains the search will be delayed by eight hours, and if it snows the search will be paused for 24 hours.

There are three local utility functions in this decision problem, namely, *Cost of mobilization (Cm)*, *Cost of search (Cs)*, and *Governmental support (Gs)*. *Governmental support* represents a monetary support which the SAR director receives to cover the expenses of a SAR mission. The government rewards the SAR director $50,000 for finding the missing person and a bonus of $50,000 if the missing person is alive when found. *Cost of search* is dependent on time in the sense that the cost increases as long as the search continues. If the person is not found within a week after the person is reported missing, it is assumed that the person will not be found, as this gives enough time to both get the heat signature and search through the entire area. At this point the active search is discontinued, and the only trace of it is a file at the SAR director's office.

The described DPIT is modelled using an IDIT, and the resulting IDIT is depicted in Figure 4.2. □

Looking at the SAR problem, we see how an IDIT represents a DPIT. The IDIT starts at the first decision node, which is Mo. Before taking the decision the decision taker observes who the missing person is. This is illustrated in Figure 4.2 by the informational arc from Mp to Mo. Furthermore, it should be noted that Mo represents an instant decision shown by the lack of an attached time node. The cost associated to mobilizing is depicted by the utility node Cm, having a functional arc into it from Mo.
The decision following *Mobilization* is the wait decision Ws, which has a time node attached in accordance to the rules of wait decisions. The node and the attached semicircle should be thought of as two separate nodes, where there exists a solid arc from the decision node to the time node. The contraction of these are due to ease of reading, as the IDIT otherwise would clutter with arcs. Ws, and the time node attached to it, illustrates how IDITs on the qualitative level handle Requirement 1 for frameworks modelling DPITs.

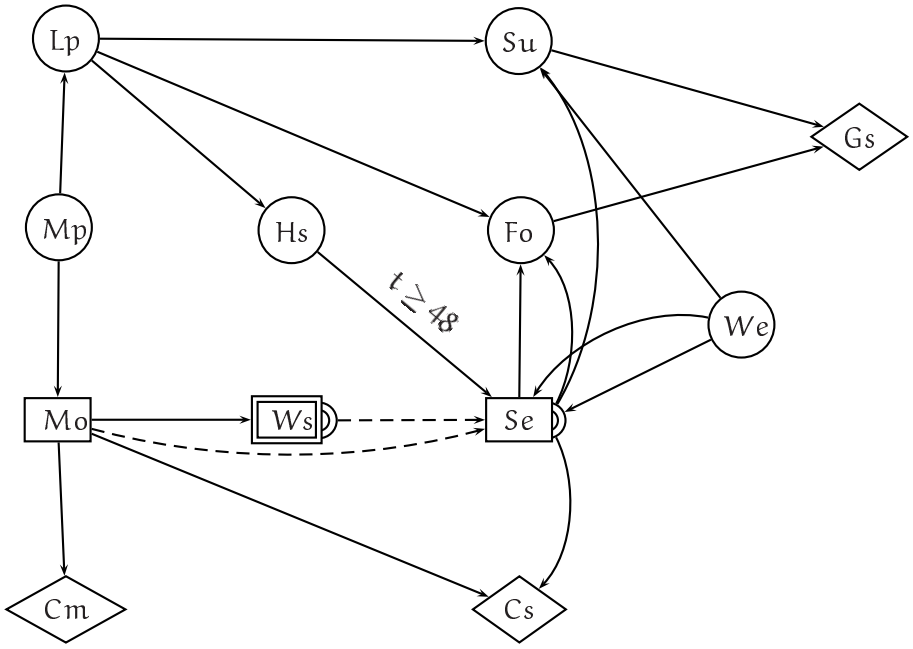


Figure 4.2: *An IDIT modelling the SAR problem.*

Continuing through the IDIT of the SAR problem the next decision node is Se, which is restricted by the choice taken in *Mobilization* and the end-time of *When to begin searching*, as shown by two the restriction arcs. These restriction arcs depict how IDITs handle Requirement 2 on the qualitative level.
Before taking the decision, the decision taker observes *Weather* and if the initiation-time is greater than or equal to 48 hours, *Heat signature* is also observed. The latter shows an example of a guarded arc, which is how IDITs satisfy Requirement 4.
Cs, which represents the cost of searching, is influenced by the choice of *Mobilization* and the end-time of *Search*. This is illustrated by having functional arcs from both Mo and the time node attached to Se going to Cs. The node is an example of a utility function dependent on time, and shows how IDITs handles Requirement 6, on the qualitative level.
The end-time of *Search* is influenced by the choice taken in *Search*, the end-time of *When to begin searching*, and *Weather*. The fact that the end-time is influenced by *Weather* can be seen directly in the IDIT by the dependence arc emanating from We going into the time node. This illustrates how IDITs handle Requirement 3. That the end-time of Se is influenced by the end-time of Ws cannot be seen in the graphical representation. This is a deliberate choice to avoid the diagram cluttering with arcs.
The remaining chance variables are never observed or observed too late to have an impact on taking decisions. Lp represents an ordinary chance variable and the arc going into Lp represents conditional dependence. Su and Fo both represent chance variables dependent on time, which is illustrated by the dependence arcs emanating

from the time node attached to Se. This illustrates how IDITs handle Requirement 5. Besides the dependence arc from the time node both have other dependence arcs going into them, illustrating the rest of their conditioning sets.

The SAR problem does not include any occurrences of post-realized utility functions or cycles. To illustrate how these are handled by IDITs we present an example we call the software release problem. We have split the example in two, one in which we discuss cycles, and one in which we discuss post-realized utility functions. The examples are presented in Example 2

Example 2
This DPIT takes outset in a software development project. We focus on two different parts of the process. The first part we look at concerns the transition from the analysis phase to the design phase, and the second part concerns what might happen after the software has been released. The two examples have been simplified for ease of understanding.

Consider a software development scenario in which the analysis is about to be concluded. The two decisions following this are one concerning prototyping and one concerning design. With an object-oriented approach, the order in which these phases is taken should not matter [Mathiassen et al., 2001]. The factor, which determines which phase to begin, could be the amount of time the analysis has taken. That is, the end-time for the analysis determines in what sequence the prototyping and the design phases are taken.

This gives three decisions for the first part of this example. These are *Analysis (An)*, *Prototyping (Pr)*, and *Design (De)*.
To keep this simple, all decision variables have a binary state space. The choices for *Analysis* are *cursory analysis* and *thorough analysis*. Choosing a cursory analysis results in the phase taking two months, while choosing a thorough analysis results in four months of work. We refrain from determining the state space of *Prototyping* and *Design*, as these have no impact on the focus of this example.

These characteristics introduce a cycle in the IDIT. In order to make it a valid cycle, there must be guards on all arcs in the cycle, and these guards must be mutually exclusive, such that any configuration of the last time variable before the cycle, breaks the cycle. Figure 4.3 depicts the resulting IDIT.

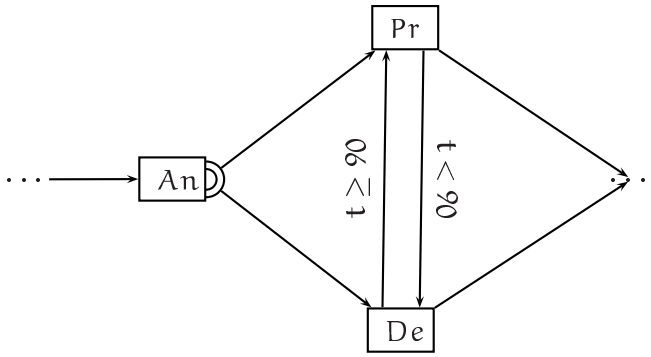


Figure 4.3: *A cycle between the decisions* Pr *and* De.

In Figure 4.3 $\cdots$ denotes the sets of variables observed or decided upon before *Analysis* and after either of *Prototyping* and *Design*, respectively.

Being able to model cycles gives the framework more expressive power. And it opens the possibility of having IDITs more correctly portray how we handle problems we are faced with in our every day life, problems similar to the one just described, where we know we have more than one thing to do, and time helps us decide in which order we do things.

The second part of this software development scenario concerns the release part of the process. The project manager has one decision to consider in this part of the software development process. This decision is *Release (Re)*, and the states of *Release* are *now*, *postpone two weeks*, and *complete missing bits*. If the software manager chooses to postpone two weeks, the time is used to either complete some missing tests or correct errors found during testing of the software.
There is one chance variable, *Faults found after installation (Fi)*, which has the states *2*, *10*, and *25*. Where each state indicates the number of errors and crashes after 100 executions of the software. This chance variable is influenced by the end-time of *Release* and a number of variables not present in this simplified DPIT.
After the software has been released the customer tests it, and if he experiences crashes of the software, or finds other faults in it, he sends back a description of these unfortunate occurrences to the software company. The software company is then obligated to correct this as best it can. The software company has no direct influence over whether the customer finds any flaws, or how long it takes before the customer contacts the software company with these. The amount of flaws has an impact on some extra expenses, which goes to wages and compensation to the customer, as he is forced to wait even longer before he can put his much needed piece of software to work. This amount of extra expenses is represented as a local utility function, which is realized at a point in time, which is later than the last decision in the DPIT.
To represent that some time passes after the software has been released, a free time variable, T', is present. This represents the time period between actually releasing the software, that is, the end-time of *Release*, and the point in time the local utility function is realized.
The local utility function for this example is named *Extra expenses*, and represents the amount of money the software company spends on correcting possible errors in the software. *Extra expenses* represents a post-realized utility function.

Figure 4.4 depicts the IDIT for the second part of this example.

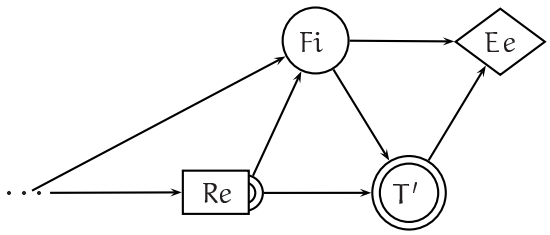


Figure 4.4: *An IDIT modelling the second part of the software development problem.*

In Figure 4.4 $\cdots$ denotes the set of variables observed or decided upon before considering *Release*.

It should be noted that there may be multiple cycles and post-realized utility functions in an IDIT, but, as per the definition of post-realized utility functions, no decisions may be present after the first occurrence of a post-realized utility function. □

The quantitative part of IDITs represents the local information relating to the individual variable or utility function. The state space of each variable is an example of the information hidden in the quantitative level of IDITs. Other than this the quantitative level consists of four sets, these being: a set of probability distributions for the chance variables; a set of local utility functions; a set of density functions for the time variables; and a set of restriction functions.

For each chance variable in the DPIT the set of probability distributions contains a conditional probability distribution for the chance variable given its parents.
The set of utility functions consists of all local utility functions. A local utility function maps each configuration of its parents into a real number representing the preferences of the decision taker.
For each time variable, T, in the IDIT, the set of density functions include a density function describing the conditional probability distribution of T given its parents as the conditioning set. The uncertainty of the time variable is shown by the degrees of freedom the density function has.
The set of restriction functions consists of all restrictive functions in the DPIT. Such functions are either related to the guards on arcs or the restriction of the state space of some decision as a consequence of the configuration of the time variable representing its initiation-time.

To give an impression of the quantitative level of an IDIT, we describe the quantitative part of the SAR problem in Example 3.

Example 3
In Example 1 the qualitative level of the SAR problem was described. The example also discussed the state space of the variables in the DPIT, which in fact is a part of the DPIT belonging to the quantitative level. We, however, choose to present the states of the variables in Example 1, to give the reader a better idea of how the DPIT is modelled, and to give the reader a more intuitive approach to the SAR problem.

IDITs represent the SAR problem on the quantitative level by four sets of functions. IDITs specify a conditional probability distribution for each chance variable given its conditioning set and a density function for each time variable. The conditioning set can be deduced from the qualitative level, as the set of parents of the chance node. One way of representing these probability distributions is by a set of tables, where each table represents the conditional probability distribution for one chance variable.

The marginal probability distribution for *Missing person* is given in Table 4.1(a). *Location of missing person* has a conditional probability distribution with a conditioning set consisting of *Missing person*, and is presented in Table 4.1(b).

The conditional probability distribution for *Heat signature* given *Location of missing person* is shown in Table 4.2(a) and Table 4.2(b) shows the marginal probability distribution for *Weather*.

lg	*am*	*em*
0.3	0.5	0.2

(a)

		Mp		
		lg	*am*	*em*
Lp	*fo*	0.8	0.55	0.2
	mo	0.2	0.45	0.8

(b)

Table 4.1: *(a): The marginal probability distribution for Missing person.(b): The conditional probability distribution for Location of missing person given Missing person.*

		Lp	
		fo	*mo*
Hs	*fo*	0.8	0.1
	mo	0.2	0.9

(a)

su	*ra*	*sn*
0.7	0.2	0.1

(b)

Table 4.2: *(a): The conditional probability distribution for Heat signature given Location of person.(b): The marginal probability distribution for Weather.*

The conditional probability distributions of the two chance variables, which are dependent on time, are described in the quantitative part of IDITs by two functions which takes a point in time, and a set of parameters, which is found using the discrete variables of the conditioning set, as their arguments, and returns the probability of the variable given the conditioning set.
Survivability has a conditioning set consisting of the discrete variables *Location of missing person* and *Weather* and the time variable $\mathsf{T}^{e}_{\mathsf{Se}}$. The function describing $\mathsf{P}(\mathsf{Su} = \mathsf{al}|\mathsf{We}, \mathsf{Lp}, \mathsf{T}^{e}_{\mathsf{Se}})$ is:

$$s(\mathsf{t}, \mathsf{c}) = (1 - \mathsf{c})^{\mathsf{t}},$$

and $\mathsf{P}(\mathsf{Su} = \mathsf{de}|\mathsf{We}, \mathsf{Lp}, \mathsf{T}^{e}_{\mathsf{Se}})$ is $1 - s(\mathsf{t}, \mathsf{c})$, where c is a parameter given by the discrete variables, and t the time given by $\mathsf{T}^{e}_{\mathsf{Se}}$. The values of c, is found in Table 4.3(a).

Thus, the probability of survival is dropping towards zero as time passes. For instance, the chance of surviving 48 hours given the missing person is in the mountains and it stays sunny, is $s(48, 0.02) = 0.38$. This function is depicted in Figure 4.5.

Found has a conditioning set consisting of the discrete variables *Location of missing person* and *Search* and the time variable $\mathsf{T}^{e}_{\mathsf{Se}}$. The probability of *found* is described by a function, f, which is given as:

$$f(\mathsf{c}, \mathsf{t}) = \mathsf{c}^{\mathsf{t}},$$

where c is given by Table 4.3(b), and t is the point in time represented by $\mathsf{T}^{e}_{\mathsf{Se}}$.

The two conditional probability distributions for *Survivability* and *Found* are examples of how IDITs represent chance variables being dependent on time, that is, how IDITs satisfy

		Lp	
		fo	*mo*
We	*su*	0.005	0.02
	ra	0.01	0.05
	sn	0.05	0.08

(a)

		Lp	
		fo	*mo*
Se	*fo*	0.03	0
	mo	0	0.05
	bo	0.03	0.05

(b)

Table 4.3: *(a): The table of parameters for Survivability given Weather, and Location of missing person. (b): The table of parameters for Found given Search, and Location of missing person.*

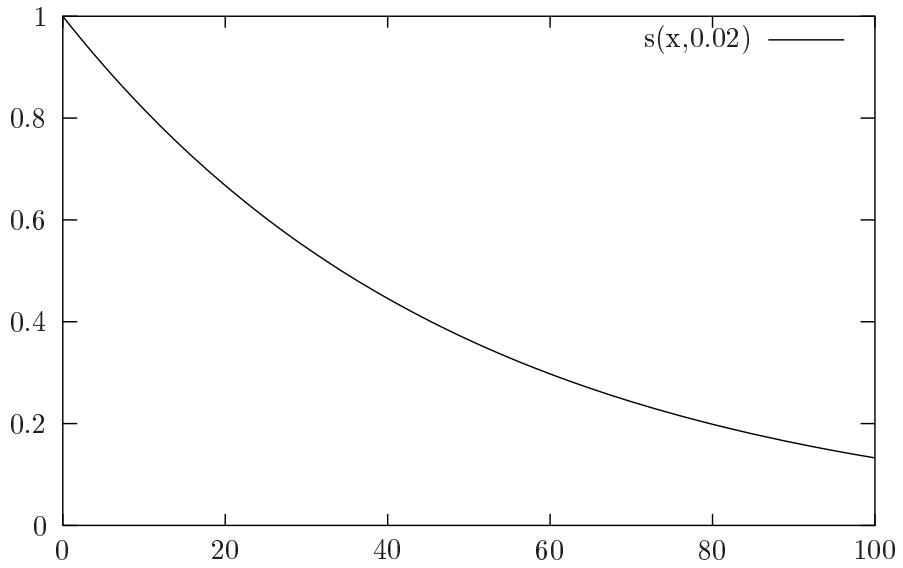


Figure 4.5: *The time dependent probability function for* $\mathsf{P}(\mathsf{Su} = \mathsf{su}|\mathsf{We}, \mathsf{Lp}, \mathsf{T}^{e}_{\mathsf{Se}})$.

Requirement 5 on the quantitative level.
The idea is to specify a function, which takes a set of arguments consisting of parameters found using the configuration of discrete parents and the point in time it is realized and mapping this to a single value, which is the probability of this exact configuration. In the cases such as the ones illustrated here, both variables have binary state spaces, which makes the summation of probabilities to one a simple task, as one state has the probability p and the other automatically has the probability $1 - p$. In cases with larger state spaces it is necessary to have a different table for each state, and the probability of each state is given by:

$$\frac{f(C = C_i, t)}{\sum_C f(C, t)}.$$

When mobilizing for a SAR mission, the SAR director pays expenses according to the choice taken at *Mobilization*. If he has chosen not to mobilize anything he pays nothing. The mobilization of SAR dogs costs \$2,000, and the mobilization of helicopters costs \$8,000. The cost of mobilizing both dogs and helicopters is the sum of those two, thus \$10,000.
The local utility function for *Cost of Mobilization* is given in Table 4.4

no	*do*	*he*	*bo*
\$0	\$2,000	\$8,000	\$10,000

Table 4.4: *The local utility function for Cost of mobilization.*

The cost of searching depends on what type of searching is initiated. Dogs cost \$100 per hour during the search plus \$1,000 at the beginning of the search period. Helicopters cost \$500 per hour of the search mission. The cost of searching with both dogs and helicopter costs an additional \$500 per day to cover expenses for communications between the two search parties.
The local utility function for *Cost of search* is a time dependent utility function, realized as three linear functions one for each state of *Mobilization*. The three functions are: for *dogs* the function is $u1(t) = 1,000 + 100t$; for *helicopters* the function is $u2(t) = 500t$; and for *both* the function is $u3(t) = 1,000(500/24 + 100 + 500)t$, and are illustrated in Figure 4.6

As can be seen from Figure 4.2 the time node, T^e_{Ws}, is only influenced by the choice at *When to begin searching*. The density function should express: zero probability states for the interval of the time variable, $[0 : ws[$, where ws is the point in time chosen at Ws; a large increase in probability immediately after the number of hours chosen in Ws, and then the probability goes towards zero after a few hours. These restrictions to the probability distribution of T^e_{Ws} are a result of the semantics and representation of time variables. If the choice at *When to begin searching* is 18 hours, the density function of T^e_{Ws} could be as depicted in Figure 4.7, with 1.50 degree of freedom.

The density function of the time variable, T^e_{Se}, associated with *Search* is influenced by T^e_{Ws}, *Search*, *Mobilization*, and *Weather*. The choices of *Mobilization*, *Search* and the true state of *Weather* implies which arguments are to be supplied to the density function. Table 4.5 illustrates these arguments. The arguments are on the form (a, b), where a is the degree of freedom, and b is the displacement on the x-axis. The displacement of some time variable, T^e_D, is determined by the end-time of the time variable in $\mathbf{pa}(T^e_D)$. In Table 4.5 entries of the form $(0, 0)$ indicate invalid configurations.

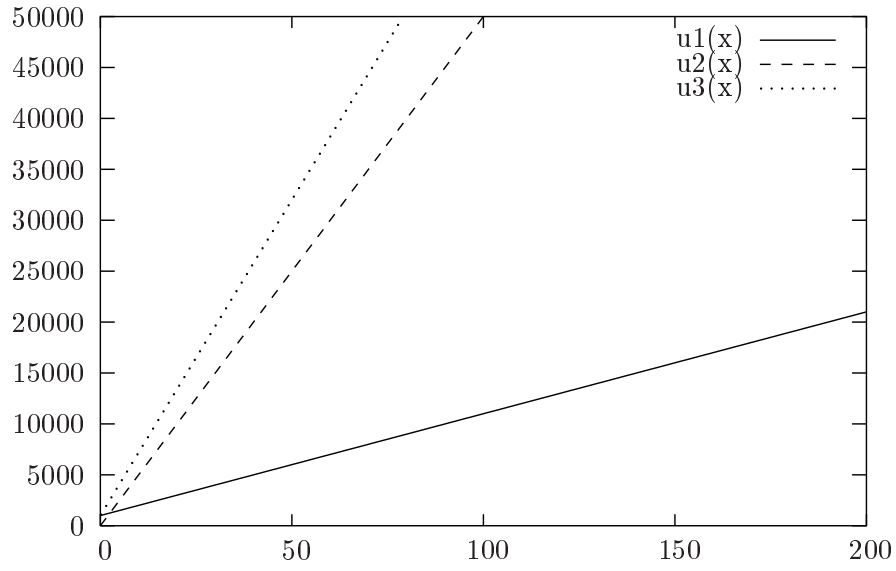


Figure 4.6: *The time dependent utility function for Cost of search.*

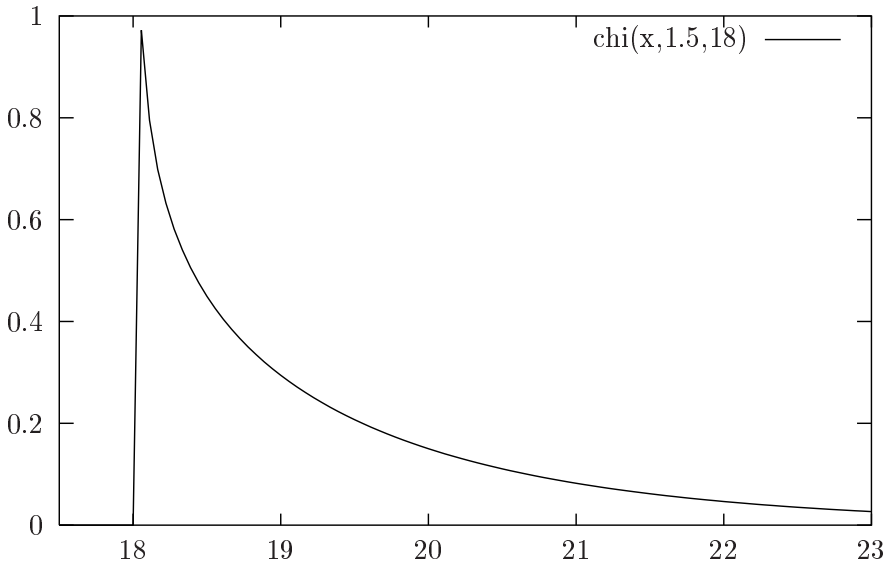


Figure 4.7: *The density function for* T^{e}_{Ws}.

		Se								
		fo			*mo*			*bo*		
		Mo			*Mo*			*Mo*		
		do	*he*	*bo*	*do*	*he*	*bo*	*do*	*he*	*bo*
We	*su*	(5.5,36)	(0,0)	(0,0)	(6.5,80)	(3,18)	(6.5,80)	(8,116)	(0,0)	(5.5,36)
	ra	(7,44)	(0,0)	(0,0)	(8.5,88)	(3.5,26)	(8.5,88)	(13,124)	(0,0)	(7,44)
	sn	(11,60)	(0,0)	(0,0)	(11,104)	(4,42)	(11,104)	(20,144)	(0,0)	(11,60)

Table 4.5: *The table of arguments for* T^{e}_{Se} *given Weather, Mobilization, and Search.*

The function for T^{e}_{Se}, given a mobilization of both dogs and helicopters; the end-time of waiting being 21 hours; and having decided to search both forest and mountains gives, dependent on the weather, one of the three density functions illustrated in Figure 4.8.

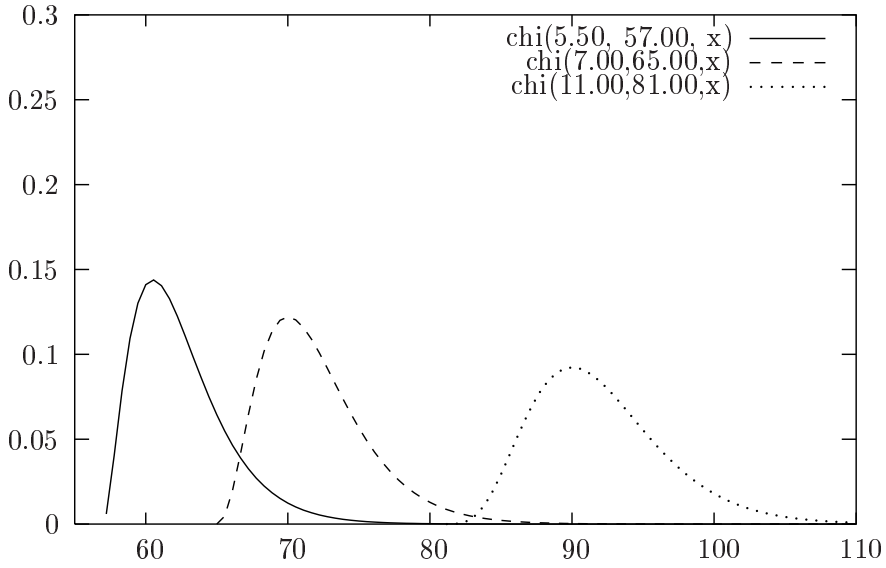


Figure 4.8: *Three functions for* T^{e}_{Se} *for different states of weather.*

This is an example of how IDITs express the uncertainty of time, in a manner satisfying Requirements 1, 3, and 5 for time variables on the quantitative level.

A constraint the SAR director must take into consideration, when deciding what to do, is the one which is imposed by his choice for *Mobilization* setting a constraint on when the searching can begin, as the SAR teams have to reach Lost Dale first. The extent of this constraint is described in Example 1.
Finally, if the SAR director wants to have a heat signature, before choosing where to search, he has to wait until the 48th hour, before taking the search decision.
In the SAR problem there is only one decision which has a restricted state space. Furthermore, there is a guard on one arc. The set of restriction functions therefore consists of these functions.
The state space of *Search* is restricted by the end-time of *When to begin searching* and the

choice of *Mobilization*. This yields the following restriction function:

$$f_{\mathsf{Se}} = \begin{cases} \mathsf{Mo} = do, \text{ and } \mathsf{T}^{e}_{Ws} < 18 \rightarrow \{no\} \\ \mathsf{Mo} = do, \text{ and } \mathsf{T}^{e}_{Ws} \geq 18 \rightarrow \{no,\ fo,\ mo,\ bo\} \\ \mathsf{Mo} = he, \text{ and } \mathsf{T}^{e}_{Ws} < 12 \rightarrow \{no\} \\ \mathsf{Mo} = he, \text{ and } \mathsf{T}^{e}_{Ws} \geq 12 \rightarrow \{mo\} \\ \mathsf{Mo} = bo, \text{ and } \mathsf{T}^{e}_{Ws} < 20 \rightarrow \{no\} \\ \mathsf{Mo} = bo, \text{ and } \mathsf{T}^{e}_{Ws} \geq 20 \rightarrow \{no,\ fo,\ mo,\ bo\ \} \end{cases}$$

The guard on the arc between Hs and Se results in the following restriction function:

$$f_{\mathsf{Se}} = \begin{cases} \mathsf{T}^{e}_{Ws} < 48 \rightarrow \mathsf{Hs} \notin \mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{Se}) \\ \mathsf{T}^{e}_{Ws} \geq 48 \rightarrow \mathsf{Hs} \in \mathbf{obs}_{\overrightarrow{\mathsf{t}}}(\mathsf{Se}) \end{cases}$$

The restriction functions for *Search* illustrate how IDITs handle Requirement 2 for frameworks modelling DPITs. □

We refrain from presenting the tables, density functions, and utility functions for the software release problem, as the information they provide is not essential in understanding how cycles and post-realized utility functions are represented in IDITs.

4.2 Formal Description of Influence Diagrams Involving Time

In this section we give a formal definition of both the qualitative and the quantitative levels of IDITs.
We begin by describing the qualitative level, in which we present a formal definition for the syntactical parts of an IDIT. In order to put this into a "real world" context, we illustrate this using the SAR problem. Following the discussion of the qualitative level, we give a definition of the quantitative level of IDITs, which is followed by discussions of the elements presented in this definition.

4.2.1 The Qualitative Level of Influence Diagrams Involving Time

The qualitative level of an IDIT consists of a labelled directed graph, which must follow a set of syntactical rules for the correlation of the elements of the graph. To put this formally we refer to Definition 4.1. The rules are discussed informally after the formal description.

Definition 4.1 (Qualitative Level of an IDIT)
Let $\mathcal{I} = (\mathbf{V}, \mathbf{L}, \mathbf{E})$ *be a labelled directed graph, with nodes* $\mathbf{V}$*, labels* $\mathbf{L}$*, and arcs* $\mathbf{E}$*. The nodes can be partitioned in six disjoint subsets:* $\mathbf{V}_{\mathsf{DC}}$*,* $\mathbf{V}^{\mathsf{e}}_{\mathsf{T}}$*,* $\mathbf{V}_{\mathsf{F}}$*,* $\mathbf{V}_{\mathsf{DD}}$*,* $\mathbf{V}_{W}$*, and*

$\mathbf{V}_{\mathsf{U}}$, *where they hold nodes representing discrete chance variables, time variables representing end-times of decisions, free time variables, discrete decision variables, wait decision variables, and local utility functions, respectively. Furthermore, the set* $\mathbf{V}_{\mathsf{DD}} \cup \mathbf{V}_{\mathsf{W}}$ *constitutes the set of decisions,* $\mathbf{V}_{\mathsf{D}}$*; the set* $\mathbf{V}_{\mathsf{T}}^{\mathsf{e}} \cup \mathbf{V}_{\mathsf{F}}$ *constitutes the set of time variables,* $\mathbf{V}_{\mathsf{T}}$*; and the set* $\mathbf{V}_{\mathsf{DC}} \cup \mathbf{V}_{\mathsf{T}}$ *constitutes the set of chance variables,* $\mathbf{V}_{\mathsf{C}}$*. The set of labels consists of functions,* $\mathsf{f} : \mathbb{R} \mapsto \{\mathsf{true}, \mathsf{false}\}$*. Furthermore, the edges can be partitioned in two disjoint sets:* $\mathbf{E}_{\mathsf{g}}$ *and* $\mathbf{E}_{\mathsf{d}}$*.* $\mathbf{E}_{\mathsf{d}}$ *is the set of restriction arcs, and are a subset of* $\mathbf{V} \times \mathbf{V}$*. And* $\mathbf{E}_{\mathsf{g}}$ *represents the set of the remaining arcs, and are a subset of* $\mathbf{V} \times \mathbf{V} \times \mathbf{L}$*. Then* $\mathcal{I}$ *is an IDIT, modelling some DPIT, if it satisfies that:*

1. *for all* V *in* $\mathbf{V}$, $|\mathbf{pa}(\mathsf{V}) \cap \mathbf{V}_{\mathsf{T}}|$ *is zero or one,*
2. *for all* V *in* $\mathbf{V}_{\mathsf{D}} \cup \mathbf{V}_{\mathsf{C}}$, $\mathbf{ch}(\mathsf{V}) \neq \varnothing$,
3. *for all* V *in* $\mathbf{V}_{\mathsf{U}}$ $\mathbf{ch}(\mathsf{V}) = \varnothing$,
4. *there is a directed path,* $\mathcal{P}$*, in* $\mathcal{I}$*, such that* $\mathbf{V}_{\mathsf{D}} \cup \mathbf{V}_{\mathsf{T}}$ *is a subset of the nodes of* $\mathcal{P}$ *for every possible configuration of the variables in* $\mathbf{V}_{\mathsf{T}}$,
5. *for all* V *and* V' *in* $\mathbf{V}_{\mathsf{T}}$*, there is a path from* V *to* V'*, if not, then there is a path from* V' *to* V,
6. *for all* $\mathsf{T}_{\mathsf{D}}^{\mathsf{e}}$ *in* $\mathbf{V}_{\mathsf{T}}^{\mathsf{e}}$ *there exists a* D *in* $\mathbf{V}_{\mathsf{D}}$*, such that* $\mathsf{T}_{\mathsf{D}}^{\mathsf{e}}$ *is in* $\mathbf{ch}(\mathsf{D})$,
7. *for all* D *in* $\mathbf{V}_{\mathsf{W}}$, $\mathbf{ch}(\mathsf{D})$ *equals* $\{\mathsf{T}_{\mathsf{D}}^{\mathsf{e}}\}$*, where* $\mathsf{T}_{\mathsf{D}}^{\mathsf{e}}$ *is in* $\mathbf{V}_{\mathsf{T}}^{\mathsf{e}}$*, and* $(\mathsf{D}, \mathsf{T}_{\mathsf{D}}^{\mathsf{e}}, \mathsf{true})$ *is in* $\mathbf{E}_{\mathsf{g}}$*, and there exists a decision* $\mathsf{D}' \in \mathbf{V}_{\mathsf{DD}}$*, such that* $(\mathsf{T}_{\mathsf{D}}^{\mathsf{e}}, \mathsf{D}', \mathsf{true})$ *is in* $\mathbf{E}_{\mathsf{g}}$.
8. *for all* T *in* $\mathbf{V}_{\mathsf{F}}$ *there does not exist a* D*, such that* D *is in* $\mathbf{de}(\mathsf{T})$,
9. *for all* (V, D) *in* $\mathbf{E}_{\mathsf{d}}$, D *is in* $\mathbf{V}_{\mathsf{D}}$*, and*
10. *for all* $\mathsf{T} \in \mathbf{V}_{\mathsf{T}}$*, there exists a* $\mathsf{T}' \in \mathbf{V}_{\mathsf{T}}$*, for which it holds that* $\mathsf{T}' \in \mathbf{pa}(\mathsf{T})$*, or* $\mathbf{V}_{\mathsf{T}} \setminus \{\mathsf{T}\} \subseteq \mathbf{de}(\mathsf{T})$.

In order to keep IDITs simple all arcs, besides restriction arcs, are associated to a label. Conventionally, we label all *unguarded arcs* with the label true.
To ease the reading of the graphical representation of an IDIT, we do not print the label the unguarded arcs. To separate the unguarded arcs from the arcs of $\mathbf{E}_{\mathsf{d}}$, we draw the arcs of $\mathbf{E}_{\mathsf{d}}$ as dashed arcs.
The rules of Definition 4.1 ensure that the directed labelled graph follows the syntax of IDITs. (1) ensures that two time nodes cannot be parent of the same node, and (2) removes the possibility of barren nodes. (3) specifies that utility nodes cannot have children. (4) secures that there exists at least one path through all time and decision variables, and (5) says that two time variables are ordered by a path between these two nodes. (6) ensures that all time variables representing end-times of decisions are associated to a decision, (7) says that wait decisions always have an end-time, and

that there exists a decision following the wait decision and (8) ensures that free time variables are only found after the last decision, which is a consequence of the no-delay assumption. (9) says that restriction arcs are only allowed into decision nodes. (10) ensures that time variables form a path, when also respecting (1).

Besides these rules the IDIT must follow the semantics of arcs and nodes as discussed in Section 4.1 and the requirements presented in Chapter 3.

To illustrate the qualitative level of an IDIT, the SAR-problem of Example 1 is presented formally.

Example 4
The graphical representation of the SAR problem can be seen in Figure 4.2. Formally, the SAR-problem is specified as below:

$\mathbf{V} = \{\mathsf{Mo}, \mathsf{Se}, \mathit{Ws}\} \cup \{\mathsf{Mp}, \mathsf{Lp}, \mathsf{Hs}, \mathsf{Fo}, \mathsf{Su}, \mathit{We}, \mathsf{T}^e_{\mathit{Ws}}, \mathsf{T}^e_{\mathit{Se}}\} \cup \{\mathsf{Cm}, \mathsf{Cs}, \mathsf{Gs}\}.$

$\mathbf{L} = \{\mathsf{true}, \mathsf{t} \geq 48\}.$

$\mathbf{E} = \{(\mathsf{Mp}, \mathsf{Lp}, \mathsf{true}), (\mathsf{Mp}, \mathsf{Mo}, \mathsf{true}), (\mathsf{Lp}, \mathsf{Hs}, \mathsf{true}), (\mathsf{Lp}, \mathsf{Fo}, \mathsf{true}), (\mathsf{Lp}, \mathsf{Su}, \mathsf{true}), (\mathsf{Mo}, \mathsf{Cm}, \mathsf{true}), (\mathsf{Mo}, \mathsf{Cs}, \mathsf{true}), (\mathsf{Mo}, \mathit{Ws}, \mathsf{true}), (\mathsf{Mo}, \mathsf{Se}), (\mathsf{Mo}, \mathsf{Se}, \mathsf{true}), (\mathsf{Hs}, \mathsf{Se}, \mathsf{t} \geq 48), (\mathsf{T}^e_{\mathit{Ws}}, \mathsf{Se}), (\mathsf{T}^e_{\mathit{Ws}}, \mathsf{Se}, \mathsf{true}), (\mathsf{Su}, \mathsf{Gs}, \mathsf{true}), (\mathsf{Fo}, \mathsf{Gs}, \mathsf{true}), (\mathsf{Se}, \mathsf{Fo}, \mathsf{true}), (\mathsf{T}^e_{\mathsf{Se}}, \mathsf{Cs}, \mathsf{true}), (\mathsf{T}^e_{\mathsf{Se}}, \mathsf{Fo}, \mathsf{true}), (\mathsf{T}^e_{\mathsf{Se}}, \mathsf{Su}, \mathsf{true}), (\mathit{We}, \mathsf{Se}, \mathsf{true}), (\mathit{We}, \mathsf{T}^e_{\mathsf{Se}}, \mathsf{true}), (\mathit{We}, \mathsf{Su}, \mathsf{true}), (\mathsf{Se}, \mathsf{T}^e_{\mathsf{Se}}, \mathsf{true}), (\mathsf{T}^e_{\mathit{Ws}}, \mathsf{T}^e_{\mathsf{Se}}, \mathsf{true}), (\mathit{Ws}, \mathsf{T}^e_{\mathit{Ws}}, \mathsf{true})\}.$

The SAR-problem satisfies all the rules of the qualitative level of an IDIT. (1), (2), (3), (9), and (10), are obviously satisfied, and (4) is fulfilled by the path $(\mathsf{Mo}, \mathit{Ws}, \mathsf{T}^e_{\mathit{Ws}}, \mathsf{Se}, \mathsf{T}^e_{\mathsf{Se}})$, and so is (5). (6) is satisfied by the arcs $(\mathit{Ws}, \mathsf{T}^e_{\mathit{Ws}}, \mathsf{true})$, and $(\mathsf{Se}, \mathsf{T}^e_{\mathsf{Se}}, \mathsf{true})$, and (8) is not applicable in the SAR-problem as there are no post-realized utility functions. The arcs $(\mathit{Ws}, \mathsf{T}^e_{\mathit{Ws}}, \mathsf{true})$, $(\mathsf{T}^e_{\mathit{Ws}}, \mathsf{Se})$, and $(\mathsf{T}^e_{\mathit{Ws}}, \mathsf{Se}, \mathsf{true})$ satisfy (7). □

Definition of a Temporal Ordering

In Chapter 3 we discussed the temporal ordering, $\rightarrow$, of a DPIT. This temporal ordering is reflected in the qualitative level of an IDIT and sets the order in which an IDIT is read.

Definition 4.2 defines the temporal ordering of the variables given time, $\prec_{\overrightarrow{\mathbf{t}}}$, for an IDIT. The temporal ordering is a total ordering of time and decision variables and a partial ordering of the set of all variables, as it does not order the discrete chance variables. We assume that, to each decision variable, D, there is a time variable, $\mathsf{T}^e_{\mathsf{D}}$, associated.
After defining the temporal order, we discuss an operational approach for finding it.

Definition 4.2 (Temporal Ordering, $\prec_{\overrightarrow{\mathbf{t}}}$)
Let $\mathcal{I} = (\mathbf{V}, \mathbf{L}, \mathbf{E})$ *be an IDIT;* $\overrightarrow{\mathbf{t}}$ *some configuration of the variables in* $\mathbf{V}^e_{\mathsf{T}} \cup \mathbf{V}_{\mathsf{T}}$*;* $\mathsf{D}_1, \mathsf{D}_2, \ldots, \mathsf{D}_n$ *an ordered sequence of nodes in* $\mathbf{V}_{\mathsf{DD}} \cup \mathbf{V}_{\mathit{W}}$*, where* D_i *is taken immediately before taking* D_{i+1}*;* $\mathbf{I}_i$ *the subset of* $\mathbf{V}_{\mathsf{DC}}$*, which is observed before* D_i*; and* $\mathbf{I}_{n+1}$ *the subset of* $\mathbf{V}_{\mathsf{DC}}$*, which is never observed, or observed too late to have an*

influence on any decision or time variable. Then the temporal order of $\mathcal{I}$ is defined as:

$$\mathbf{I}_1 \prec_{\mathbf{t}}^{\rightarrow} \mathsf{D}_1 \prec_{\mathbf{t}}^{\rightarrow} \mathsf{T}^{e}_{\mathsf{D}_1} \prec_{\mathbf{t}}^{\rightarrow} \cdots \mathbf{I}_n \prec_{\mathbf{t}}^{\rightarrow} \mathsf{D}_n \prec_{\mathbf{t}}^{\rightarrow} \mathsf{T}^{e}_{\mathsf{D}_n}$$
$$\mathbf{prior}(\mathsf{T}_1) \prec_{\mathbf{t}}^{\rightarrow} \mathsf{T}_1 \cdots \mathbf{prior}(\mathsf{T}_n) \prec_{\mathbf{t}}^{\rightarrow} \mathsf{T}_n \prec_{\mathbf{t}}^{\rightarrow} \mathbf{I}_{n+1},$$

where $\mathsf{T}^{e}_{\mathsf{D}_i}$ are time nodes representing end-times for decisions and T_i are free time nodes.

The temporal order as shown in the definition is found by identifying the first decision of the IDIT; identifying its set of observed variables and its associated time node, if it has any; and then ordering these accordingly. For any time node found, it must be configured to some state, as to resolve any possible guards affecting the set of observed variables for the next decision. For all time nodes, the configuration of the last time node must be remembered and taken into consideration. Then the ordered nodes are removed and the operation is repeated until no more decisions exist. If, after this, there are more time nodes, the first of these is identified in a similar manner as used in identifying the first decision node. The prior of the time node is identified and the nodes are ordered accordingly. The ordered nodes are removed and the process continues until no more time nodes exist. Then the remaining nodes are placed in the set we have chosen to call $\mathbf{I}_{n+1}$.

Definition 4.2 relies on there being a decision, which is before all others. Theorem 4.3 ensures that this decision exists and shows how it is found.

Theorem 4.3 (First Decision)
In an IDIT, $\mathcal{I}$, there exists a decision, D, such that $\mathsf{D} \prec_{\mathbf{t}}^{\rightarrow} \mathsf{D}'$ for all D' in $\mathbf{V}_{\mathsf{D}} \setminus \{\mathsf{D}\}$. This is the decision, D, which for any $\overrightarrow{\mathbf{t}}^{\downarrow \varnothing}$ has no other decisions as ancestor.

Proof: Assume that such a first decision is not unique. Then two or more first decisions would exist, and, as IDITs require, there is a path between them, either, one would be before all others, contradicting there being multiple first decisions, or, there is a cycle between them. If such a cycle exists there needs to be guards on the arcs of the cycle, such that a configuration of the time nodes would render the diagram acyclic. As guards are only allowed to reference time variables before the cycle, and time variables are associated to decisions, some decision must be before the two first decisions, contradicting that they are the first decisions. Furthermore, if a free time variable exists, then that time variable must be after the last decision, and therefore also after any decision in a cycle. This proves that only one first decision exists, and it can be uniquely identified. ■

It should be clear that Definition 4.2 does not imply there only being one temporal ordering for an IDIT. In fact several temporal orderings can exist, depending on the different configurations of time variables. This indicates that an infinite number of orderings exists on the quantitative level, however, as IDITs require that there

needs to be a finite number of variables, thus also a finite number of guards, some configurations yield equivalent temporal orderings. Later we describe how to find the number of temporal orderings necessary to have a welldefined IDIT, both for the qualitative and the quantitative level.
The temporal orderings of an IDIT are defined through the semantics of the arcs and nodes on the qualitative level, and the semantics of a set of restriction functions found on the quantitative level, that is, restrictions imposed by guards and restriction arcs. A temporal ordering is built such that it follows the manner of reading an IDIT. This means that, when reading an IDIT, if some variable, V, is read before another variable, V', V comes before V' in the temporal order, denoted as $V \prec_{\mathbf{t}}^{\rightarrow} V'$. If V is read immediately before reading V', then there is no node, V'', such that $V \prec_{\mathbf{t}}^{\rightarrow} V'' \prec_{\mathbf{t}}^{\rightarrow} V'$.
We have defined a temporal ordering, through how the IDIT is read. This leads to finding one temporal ordering for an IDIT. In the following section we describe how to deduce any order of an IDIT given any configuration of the time variables by using a structure we have named a *preliminary temporal ordering*.

Preliminary Temporal Ordering

The preliminary temporal ordering is a partial ordering of all nodes in an IDIT. It is the ordering, which is found without considering a configuration of the time nodes of the IDIT. This means that, when all temporal orderings are to be found, the preliminary temporal ordering is used, instead of going through the ordering of all nodes for every unique configuration of time variables. This results in the need to only go through the nodes of which the initial ordering is uncertain, as these are explicitly identified. Furthermore, this information is used when solving an IDIT, as is described in Chapter 5.

A partial temporal ordering of all variables can be deduced directly from the qualitative level of any IDIT. This orders all variables, which can be ordered, that is, if an unguarded path between two decisions exists, then the two decisions are ordered with respect to each other. Furthermore, each decision, D, is ordered with respect to the set of chance variables observed before taking D and the possible time variable associated to D. We call this ordering a preliminary temporal ordering, denoted as $<$, as it does not necessarily impose a total ordering of decisions, such as the temporal ordering, but can be deduced from the qualitative level alone.
As the preliminary temporal ordering is deduced from the qualitative level of the IDIT, guarded arcs are not evaluated. Therefore, multiple instances of a guarded variable exists in the ordering. For instance, if a chance variable, C, has a guarded arc to a decision, D, then C is both before and after D in the preliminary temporal ordering. This should be interpreted as a preliminary uncertainty on when C is observed rather than C actually being observed twice.

The preliminary temporal ordering is deduced from the IDIT by identifying the first

decision, which can be unambiguously identified, as stated by Theorem 4.3. We find the first decision by exploiting that its set of ancestors intersected with decision variables is the empty set.
In order not to confuse an observed variable with a guarded chance variable, we construct, for each decision, a set consisting of the guarded chance variables. We call this the set of *guarded observed variables*.

When the first decision is identified the set of observed variables for this decision and the time variable associated with it, are ordered with respect to the decision. That is, $\mathbf{I}_\mathsf{D} < \mathsf{D} < \mathsf{T}^e_\mathsf{D}$, where $\mathbf{I}_\mathsf{D}$ is the set of observed variables. Notice that there cannot be any guarded arcs among the arcs between $\mathbf{I}_\mathsf{D}$ and D as there is no time variable, which can be referenced.
Whenever a decision is ordered in the preliminary temporal ordering, the decision or decisions immediately following this decision, are identified. One manner of doing this is by comparing the past of all unordered decisions. The decision, which has only the set of already ordered decisions in its set of ancestors, is the next decision. If two decisions both satisfy this requirement, the two decisions are not ordered with respect to each other. Whenever there are more than two decisions, which have the same set of ancestors, multiple cycles exist. For each decision found this way the set of observed variables is identified and so is the time variable, which is associated with it.
Cycles are allowed in IDITs, if guards secure that they are broken whenever the time variables before cycles are configured. As a result of this, multiple cycles can exist, such that multiple decisions can have equivalent sets of ancestors, but there still exists an ordering of a subset of these decisions. Figure 4.9 illustrates the decisions of an IDIT in which four decisions, D_2, D_3, D_4, and D_5 have equivalent sets of ancestors, as the decisions are also treated as ancestors of themselves.

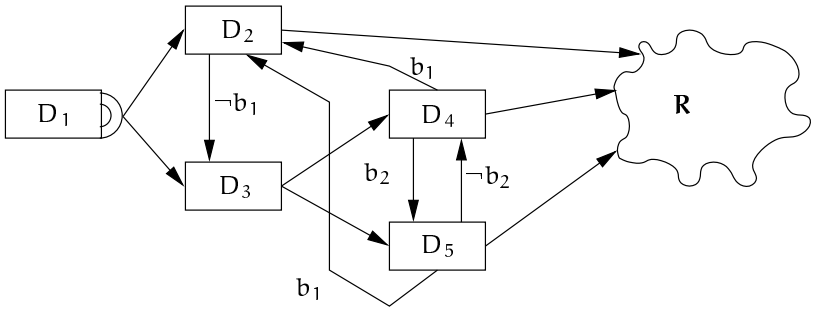


Figure 4.9: *Part of an IDIT with cycles.*

As can be deduced from Figure 4.9 the four decisions $\mathsf{D}_2, \mathsf{D}_3, \mathsf{D}_4$, and D_5 have equivalent sets of ancestors, when not considering the guards on the arcs. Considering the guards, we see that, in the preliminary temporal ordering, D_3 should be before both D_4 and D_5, and D_1 should be before all others. $\mathbf{R}$ is the set of nodes, which are

observed or taken later than those specifically represented in the figure.
Whenever two or more decisions have an equivalent set of ancestors, the set of ancestors from unguarded arcs is used to order the variables. The set of ancestors from unguarded arcs for some variable, V, is the set of variables, from which there exists a path consisting of unguarded arcs going to V, for instance, D_3 and D_1 are in the ancestor set from unguarded arcs for D_4, but D_5 is not.
If the sets of ancestors are still equivalent the decisions are entered in the preliminary temporal ordering in an unordered fashion. Each of the unordered decisions are ordered with respect to its set of observed variables and to a possible time variable associated with it.
In cases where the sets of ancestors from unguarded arcs differ for one or more decisions, at least one of the sets of ancestors from unguarded arcs include one of the other decisions being ordered. The preliminary temporal ordering, in such cases, is that a decision, which are in the set of ancestors from unguarded arcs, is placed before the other decision. For instance, in the case shown in Figure 4.9 the preliminary temporal ordering, $<$ is:

$$D_1 < T^e_D < \{D_2, (D_3 < \{D_4, D_5\})\} < \mathbf{R}.$$

One manner of reading an IDIT is to read it according to the preliminary temporal ordering. That is, to identify the first decision, then to establish, which chance variables are observed before taking this decision. When the first decision is found the next decision can be found in a manner similar to the one described above. Guarded arcs should be read as possible informational arcs, as the existence of the arc cannot be established, based only on the qualitative level. However, it gives the reader of the IDIT a sense of when the chance variable can be observed.
The preliminary temporal ordering, $<$, is related to the temporal ordering, $\prec^{\rightarrow}_{\mathbf{t}}$, such that, if two decision or time variables, V and V', are ordered as $V < V'$, then $V \prec^{\rightarrow}_{\mathbf{t}} V'$.

Deducing the preliminary temporal ordering for the SAR problem yields the following ordering:

$$Mp < Mo < Ws < T^e_{Ws} < \{Hs, We\} < Se < T^e_{Se} < \{Lp, Su, Fo, Hs\}$$

After having defined the qualitative level of IDITs, we now define the quantitative level of IDITs.

4.2.2 The Quantitative Level of Influence Diagrams Involving Time

The quantitative level of an IDIT is defined as a *realization*, given in Definition 4.4. After presenting the definition we informally discuss the implication of each syntactical rule of the realization of an IDIT. Finally, the elements of such a realization are discussed. Definition 4.4 defines an ideal realization, and as such does not discuss the problems imposed by the rules.
We use $\mathbf{pa}_{\mathrm{d}}(D)$ to denote the set of parents for some decision, D, from dashed arcs.

Definition 4.4 (Realization of an IDIT)
Let $\mathcal{I} = \{\mathbf{V}, \mathbf{L}, \mathbf{E}\}$ be the qualitative part of an IDIT modelling some DPIT. Then a realization of $\mathcal{I}$ consists of four sets of functions: $\mathbf{\Phi}$, which is the set of conditional probability distributions for chance variables in $\mathcal{I}$; $\mathbf{\Psi}$, which is the set of local utility functions for $\mathcal{I}$; $\mathbf{\Pi}$, which is the set of density functions for time variables in $\mathcal{I}$; and $\mathbf{\Gamma}$, which is the set of restriction functions associated with $\mathcal{I}$. Such that,

1. *If $C \in \mathbf{V}_C$ is in $\mathcal{I}$, then there exists a conditional probability distribution for C, $P(C|\mathbf{pa}(C))$, in $\mathbf{\Phi}$.*
2. *If $V \in \mathbf{V}_U$ is in $\mathcal{I}$, then there exists a local utility function for V, $\psi : \mathbf{sp}(\mathbf{pa}(V)) \mapsto \mathbb{R}$, in $\mathbf{\Psi}$.*
3. *T^e_D is in $\mathbf{V}^e_T$, D is in $\mathbf{V}_{DD}$, and there does not exist a time node, $T^e_{D'}$, in $\mathbf{V}^e_T$, such that T^e_D is in $\mathbf{ch}(T^e_{D'})$, iff a density function, $f_{T^e_D}(\mathbf{pa}(T^e_D))$, is in $\mathbf{\Pi}$, such that $f_{T^e_D}(\overrightarrow{c}, t)$ is zero for all configurations, $\overrightarrow{c}$, of $\mathbf{pa}(T^e_D)$ and all times, t, in $\mathbb{R}$, where t is less than zero,*
4. *T^e_D is in $\mathbf{V}^e_T$, D is in $\mathbf{V}_{DD}$, and there exists a time node, $T^e_{D'}$, in $\mathbf{V}^e_T$, such that T^e_D is in $\mathbf{ch}(T^e_{D'})$, iff a density function, $f_{T^e_D}(\mathbf{pa}(T^e_D))$, is in $\mathbf{\Pi}$, such that $f_{T^e_D}(\overrightarrow{c}, t', t)$ is zero for all configurations, $\overrightarrow{c}$, of $\mathbf{pa}(T^e_D) \setminus \{T^e_{D'}\}$ and all times, t and t', in $\mathbb{R}$, where $t \leq t'$,*
5. *T^e_D is in $\mathbf{V}^e_T$, D is in $\mathbf{V}_W$, and there does not exist a time node, $T^e_{D'}$, in $\mathbf{V}^e_T$, such that T^e_D is in $\mathbf{ch}(T^e_{D'})$, iff a density function, $f_{T^e_D}(\mathbf{pa}(T^e_D))$, is in $\mathbf{\Pi}$, such that $f_{T^e_D}(\overrightarrow{c}, d, t)$ is zero for all configurations, $\overrightarrow{c}$, of $\mathbf{pa}(T^e_D) \setminus \{D\}$ and all times, t and d, in $\mathbb{R}$, where $t \leq d$,*
6. *T^e_D is in $\mathbf{V}^e_T$, D is in $\mathbf{V}_W$, and there exists a time node, $T^e_{D'}$, in $\mathbf{V}^e_T$, such that T^e_D is in $\mathbf{ch}(T^e_{D'})$, iff a density function, $f_{T^e_D}(\mathbf{pa}(T^e_D))$, is in $\mathbf{\Pi}$, such that $f_{T^e_D}(\overrightarrow{c}, d, t', t)$ is zero for all configurations, $\overrightarrow{c}$, of $\mathbf{pa}(T^e_D) \setminus \{D, T^e_{D'}\}$ and all times, t, d, and t', in $\mathbb{R}$, where $t \leq t' + d$,*
7. *T_i is in $\mathbf{V}_T$, and T' is in $\mathbf{V}_T$, such that T_i is in $\mathbf{ch}(T')$, iff a density function $f_{T^e_D}(\overrightarrow{c}, t', t)$ is zero for all configurations, $\overrightarrow{c}$, of $\mathbf{pa}(T^e_D) \setminus \{T^e_{D'}\}$ and all times, t and t', in $\mathbb{R}$, where $t \leq t'$,*
8. *If D is in $\mathbf{V}_D$, $T^e_{D'}$ is in $\mathbf{V}^e_T$, and there is an arc, (V,D), in $\mathbf{E}_d$, then there is a restriction function for D, $f_D : \mathbf{sp}(\mathbf{pa}_d(D)) \rightarrow \mathbf{sp}(D)$, in $\mathbf{\Gamma}$, and*
9. *If (V, D, g) is in $\mathbf{E}_g$, then $T^e_{D'}$ is in $\mathbf{V}^e_T$, such that $(T^e_{D'}, D, \mathtt{true})$ is in $\mathbf{E}_g$, and g is defined as $g : \mathbf{sp}(T^e_{D'}) \mapsto \{\mathtt{true}, \mathtt{false}\}$ is in $\mathbf{\Gamma}$.*

Rule (1) determines that all chance variables in a realization of an IDIT have a conditional probability distribution attached. And the conditional probability distribution of each chance variable is in $\mathbf{\Phi}$. Rule (2) handles $\mathbf{\Psi}$, which is the set of local utility functions. It says that for each local utility in an IDIT there exists a

utility function for each configuration of parents of the local utility. These utility functions are all in Ψ. Rules (3), (4), (5), and (6) are related. The two first of these handle time variables associated with decisions, and the two last handle time variables associated with wait decisions. They all say that all time variables, T, in $\mathbf{V}_T^e$ have a density function associated with it, and the point in time represented by the individual time variable is influenced by the choice of the decision it is associated with, and the point in time represented by the time variable it has as a parent. The first time variable is, of course, only influenced by the decision it is associated with, in this respect. All the density functions are placed in Π. The rules also determine that time may not be negative and time always progresses through an IDIT. Rule (7) says the same as the four previous rules, only for free time variables. That is, the free time variables may not be negative and time progresses. And the density function associated to each free time variable takes the parents and a possible previous time variable as parameter. These density functions are also in Π. Rule(8) says that for any restriction arc in an IDIT, there exists a restriction function, which determines the state space of the restricted decision, given the state space of the restricting variable. As per our definition, only decision variables may be restricted, and only the effect of time may restrict these. All restriction functions related to restriction arcs are in Γ. Rule (9) says that for each guarded arc into a decision, there exists a time variable representing the initiation-time of that decision. The point in time this time variable represents determines if the guard is evaluated to `true` or `false`. This information is also in Γ.

We discuss, for each set of functions, which limitations are imposed in this thesis and what the consequences of this are.

Φ: This set consists of the conditional probability distributions for chance variables in an IDIT. For a discrete chance variable, C, which only has other discrete variables in its conditioning set, the conditional probability distribution associated to it is $P(C|\mathbf{pa}(C))$, which is similar to the conditional probability distributions of chance variables in influence diagrams.
When C has a time variable in its conditioning set, the conditional probability distribution cannot be defined by associating a specific probability for each state, given all configurations of the parents, as there are an infinite number of such configurations. Instead the conditional probability distribution is defined by associating to each state of C, given its discrete parents, a function over time, described by $f : \mathbb{R} \mapsto [0; 1]$. f is based on the configuration of discrete parents of C. The introduction of functions to describe the probability distributions does not alter the requirement that the probability distribution sums to one, for all configurations of parents. One manner of ensuring this is by normalizing the function. That is, a function of a state c_i is given as:

$$f_{C=c_i}(t) = \frac{f(x_i, t)}{\sum_{j=1}^{n} f(x_j, t)},$$

where x_i is a parameter for $C = c_i$ given a configuration of the discrete parents

of C, when C has n states. In cases where C is binary, the probabilities can be found as f for one state and $1 - f$ for the other.
In order to ease the development of a solution method, we further limit the functions to be those which are differentiable and integratable, as these are nice properties to have fulfilled for continuous functions.

Ψ: The set of local utility functions consists of a utility function for each utility node in the IDIT. The function maps each configuration of the parents of the utility node to a real value. When a local utility function is dependent on time a method similar to the one used for chance variables dependent on time is used, that is, the discrete parents are used to look up some parameter for a function over time. Such a function reflects the preferences of the decision taker, like for local utility functions, which are not dependent on time, and the axioms of utilities, as described in [Pearl, 1988] should also be followed. Finally, these function should be defined for all positive reals, such that they are defined for all points in time.

Π: The uncertainty associated with time variables is represented through density functions. We have chosen to have time variables be represented by χ^2-distributions. This choice is based on the semantics of time variables, namely that they portray an unforeseen delay in the end-time of some timed action, and, a χ^2-distribution most accurately portrays the intuitive conception of this semantics. This is because the output of such a distribution is a function, for which the density of the function is concentrated on the first part of the domain. When combining this with the semantics of a time variable, this is interpreted as: there is a high probability that the timed action being delayed is delayed with a short amount of time, and a very low probability of the delay being higher than some set time, dependent on the specific parameters for the χ^2-distribution of the time variable at hand. Time variables, which are dependent on discrete variables are associated to a table, from which the parameters for the χ^2-distribution is found, based on a configuration of the parents. When a time variable, T, is dependent on another time variable, T', $P(T|T')$, we assume that this can be rewritten to $P(T)+t'$, where $T' = t'$, that is, $P(T|t') \sim P(T)+t'$. Even though we have chosen to represent the probability distributions of time variables using the χ^2-distributions another choice of density function would not change the framework, considerably.

Γ: This defines the set of restriction functions. If the decision being restricted is a wait decision, then the state space is altered, so that points in time in specific intervals are invalid choices for that decision. If, on the other hand, the decision is a discrete decision, then certain states are restricted, that is, they are invalid. This set also handles the functions for determining if some guard evaluates to *true* or *false*.

The quantitative level of an IDIT is specified by using tables and functions as in Example 3.

The temporal ordering, as defined in Definition 4.2, is based on IDITs, in which all asymmetries have been resolved through configurations of the time variables. To resolve these asymmetries we convert the IDIT to a number of symmetric IDITs and through these we determine a temporal ordering.

4.2.3 Symmetric Influence Diagrams Involving Time

From the qualitative level of an IDIT a preliminary temporal ordering was deduced. To deduce the temporal ordering of an IDIT the quantitative level is also needed. Two decisions, which are not ordered in the preliminary temporal ordering, should be ordered in the temporal ordering. Likewise, the time of observation of guarded chance variables should be pinpointed. To find the ordering of, for instance, two such decisions, the time variable, to which their ordering refers, is identified. The state space of the time variable is divided into the values yielding one ordering and the values which yield the other ordering. Generally, we say that a time variable *splits* the IDIT into a set of new IDITs, in which the asymmetries imposed by the time variable are resolved. We sometimes refer to the time variable as a *split variable*. An IDIT, in which all split variables have been split upon, resulting in the resolution of all asymmetries, is called a *symmetric IDIT*, and, if only some of the asymmetries have been resolved, we call it a *partially symmetric IDIT*. As symmetric IDITs are special cases of partially symmetric IDITs, we only define partially symmetric IDITs. Partially symmetric IDITs are defined in Definition 4.5.

When splitting the IDIT into a set of symmetric or partially symmetric IDITs, the sets of arcs are altered. Consider an IDIT, for which, at some point, a time variable, T_n, induces a split. If the IDIT is split on T_n it means that there is a path between T_n and at least one decision, D. The set of guarded arcs going into D from variables, $V_i \in \mathbf{V}$, now become informational arcs if their guard is satisfied. The set of restriction arcs going into D are also converted, such that the state space of D is resolved to a specific state space. All other arcs remain the same. Furthermore, the set of restriction functions alters to accommodate the changes in the set of arcs.

In Definition 4.5 we use $\mathbf{E}_{d_{T_n}}$ to denote the set of dashed arcs in the partially symmetric IDIT resulting from a split on T_n. And $\mathbf{E}_{g_{T_n}}$ is the set of labelled arcs in the partially symmetric IDIT resulting from a split on T_n. If some index, T_{n-1}, does not refer to any time variable, it simply means the initial IDIT, and if some index, T_{n+1}, does not refer to any time variable, it simply means the rest of the IDIT.

Definition 4.5 (Partially Symmetric IDIT)
Let $\mathcal{I} = (\mathbf{V}, \mathbf{L}, \mathbf{E})$ *be an IDIT, and* T_n *some split variable. Then* $\mathbf{E}_{g_{T_n}} = \{(V, D_i, l) | l(T_{n-1}) = \mathsf{true}, D_i \prec^{\rightarrow}_{\mathbf{t}} T_{n+1}, (V, D_i, l) \in \mathbf{E}_{g_{T_{n-1}}}\} \cup \{(V, D_i, l) | T_{n+1} \prec^{\rightarrow}_{\mathbf{t}} D_i\}$ *and* $\mathbf{E}_{d_{T_n}} = \mathbf{E}_{d_{T_{n-1}}} \setminus \{(V, D_i) | T_{n+1} \prec^{\rightarrow}_{\mathbf{t}} D_i\}$. *Then*

$\mathcal{I}' = (\mathbf{V}, \mathbf{L}, \mathbf{E}')$ *is the partially symmetric IDIT resulting of a split on* T_n, *where* $\mathbf{E}' = \mathbf{E}_{g_{\mathsf{T}_n}} \cup \mathbf{E}_{d_{\mathsf{T}_n}}$.

Besides this qualitative definition of a partially symmetric IDIT, the set of rules, which define an IDIT, must also be followed when constructing a partially symmetric IDIT. A symmetric IDIT is a partially symmetric IDIT in which $\mathbf{E}_d$ is empty and the label on all arcs in $\mathbf{E}_g$ is true.

For the realization of a partially symmetric IDIT, we have that the set of restriction functions is changed, while everything else remains the same. That is, the set of restrictions resulting from a split on T_n is denoted as Γ_{T_n} and is defined as $\Gamma_{\mathsf{T}_n} = \{\gamma(\mathsf{D}_i) | \mathsf{T}_{n+1}\mathsf{D}_i\}$.

A symmetric IDIT has some properties, which neither partially symmetric IDITs nor IDITs have. In a symmetric IDIT there exists a total ordering of decision and time variables, and a partial ordering of all decision and chance variables. Furthermore, for each discrete chance variable it can be deduced before which decision, if ever, it is observed. From the variables of a symmetric IDIT the temporal ordering, as defined in Definition 4.2, can be deduced.
Partially symmetric IDITs have the property that all variables before the time variable introducing the next split, can be ordered according to the temporal ordering. Thus, in the temporal ordering, of partially symmetric IDITs, the set of time variables, which constitutes $\{\mathsf{T}_1, \ldots, \mathsf{T}_n\}$, is the variables before the split variable, T_n. The part of the partially symmetric IDIT following the split variable can only be ordered according to a preliminary temporal ordering, thus, some variables might not be ordered. It should be noticed that this fact also holds for IDITs, which have not been split, as variables before the first split variable are ordered in accordance to the temporal ordering.

Due to the restriction of only allowing asymmetries arising from time variables, only time variables can be split variables. A time variable splits an IDIT, or a partially symmetric IDIT, whenever a guard or restriction function is referring it. We postpone the details of the process of finding the parts, into which an IDIT is split, until Chapter 5.
The splitting of an IDIT can be illustrated by constructing a tree structure, called a *split tree*, which reveals the asymmetries imposing a split on its branches and the resulting partially symmetric and symmetric IDITs as its nodes. The root of the tree is the original IDIT, all internal nodes are partially symmetric IDITs, and all leaf nodes are symmetric IDITs. For instance, splitting the IDIT of Section 4.1.1, with respect to T^e_{Ws}, results in the split tree of Figure 4.10.

A time variable splits an IDIT if two different configurations of the variable either cause a guard to change its value, or a decision to change its state space. For instance, the instantiation of T^e_{Ws} in Example 1 will, according to the interval it is instantiated to, impose a different state space for *Search*, and change whether or not *Heat signature* can be observed before taking *Search*.

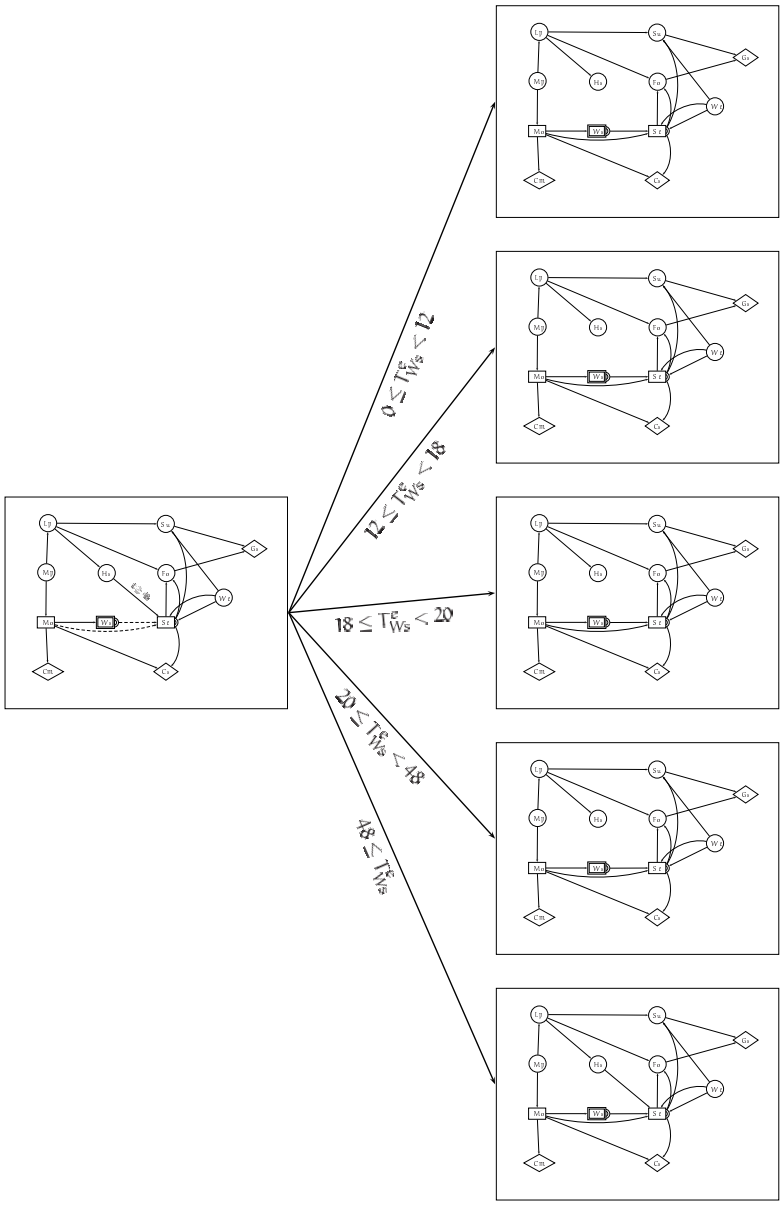


Figure 4.10: *A split tree resulting from a split on* T^e_{Ws}.

The split tree should be read such that, if T^{e}_{Ws} splits to a point in time, which is either 48 hours or later, then the temporal order of the IDIT follows how the IDIT in the lower leaf is read. Otherwise the temporal order follows how the IDIT in one of the other leaves is read. The restrictions to the state space of *Search* can only be seen in the quantitative level of the IDIT, but the fact that there are leaves with similar symmetric IDITs in them indicates that some decision variable after the last split variable has a restricted state space.
In partially symmetric and symmetric IDITs, restriction arcs referencing time variables, which have already been split upon, are exchanged with informational arcs. This we do as all information regarding the restriction has been removed, because the time variable referred by the restriction function is known to be in a specific interval. This interval is based on the restriction functions themselves. The same is true for guarded arcs. That is, if a guard is evaluated to true because of a split, the arc is present as an informational arc in the resulting partially symmetric or symmetric IDIT. If, on the other hand, the guard evaluates to false, the arc is removed. As an example of this consider Figure 4.10. When comparing the labels on the edges between the root and the leaves of the split tree, we see that only in the lower right leaf is the former guarded arc present.

The temporal ordering of a symmetric IDIT can be found by the same technique for the preliminary temporal ordering. That is, we find the first decision, D, and the chance variables observed before taking it, $\mathbf{I}_{\mathsf{D}}$, and structure those as $\mathbf{I}_{\mathsf{D}} \prec^{\rightarrow}_{\mathsf{t}} \mathsf{D}$. This is followed by the time variable associated with D. Then all variables, which have been ordered, are removed from consideration, and the process is repeated, until all variables are ordered.

The difference in the temporal ordering between the qualitative level and the quantitative level lies in the number of possible orderings. The number of temporal orderings in the qualitative level is only affected by the number of guards in the IDIT. As long as there is at least one time variable in the IDIT, there is an infinite amount of temporal orderings on the quantitative level of the IDIT.
We say that the set of temporal orderings on the qualitative level constitutes the required set of temporal orderings for an IDIT, while the set of temporal orderings on the quantitative level constitutes the set of possible temporal orderings.

4.2.4 Welldefined Influence Diagrams Involving Time

When modelling or solving IDITs it is important that the IDIT is welldefined. By welldefined we mean that the next decision to be taken can be uniquely and unambiguously identified. This requirement is satisfied as a configuration of time variables until some time variable, T_n, resolves all asymmetries between T_n and T_{n+1} in the preliminary temporal ordering, where T_{n+1} is the time variable immediately after T_n. Thus, the next decision can be found.
That welldefined IDITs are only a matter of finding the next decision is a conse-

quence of Rule (4) in Definition 4.1. For further discussion of welldefined influence diagrams, which are related to this, we refer to [Nielsen and Jensen, 1999].

4.3 Summary

In this chapter we have introduced a framework for modelling DPITs. The syntax of the framework has been defined and semantics of the concepts have been given. Furthermore, heuristics for reading and understanding an influence diagram involving time, both on the graphical level and the numerical level, have been given. This is done through the temporal and the preliminary temporal ordering. We have also argued for IDITs being welldefined.

Chapter 5

Solving Influence Diagrams Involving Time

In Chapter 4 we presented a representation language, IDIT, for modelling DPITs. In this chapter we present a solution method for solving IDITs. The solution method proposed solves IDITs with respect to finding the choice, which is preferred by the decision taker, for each decision, assuming all future choices are taken in this manner, too.
First, we present an outline of the solution method, giving an overview of the key parts of the solution. Following the overview, each part of the solution method is discussed in detail, and the difficulties of each part are identified, and, finally, in Section 5.6 the full solution method is presented. We end this chapter by, in Section 5.7, discussing sampling and the technique we have chosen.

5.1 Overview of the Solution Method

We begin with a preliminary discussion of the solution method for IDITs. It is preliminary as it does not include how to solve the individual steps of the solution method, but rather gives an overview of the steps in the solution method.
The solution method follows the solution sketch of [Broe et al., 2003]. However, where the sketch only showed how to solve a single specific example of a DPIT, the method in this section is a general solution method, such that it can be used to solve all IDITs.

A solution method of an IDIT is basically the task of determining a *policy* for each decision in the IDIT. A policy, $\delta_{\mathsf{D}}^{\prec_{\mathbf{t}}^{\rightarrow}}$, for a decision, D, given the past with respect to the temporal ordering, $\prec_{\mathbf{t}}^{\rightarrow}$, is some choice of D based on its past. The past of a decision is the set of variables, which are before the decision in the temporal ordering, that is, $\mathbf{past}(\mathsf{D})^{\prec_{\mathbf{t}}^{\rightarrow}} = \{V | V \in \mathbf{V}_{\mathsf{D}} \cup \mathbf{V}_{\mathsf{C}} \text{ and } V \prec_{\mathbf{t}}^{\rightarrow} \mathsf{D}\}$. Definition 5.1 defines a policy:

Definition 5.1 (Policy, $\delta^{\prec\overrightarrow{\mathsf{T}}}$)
Let D *be a decision in an IDIT and* $\mathbf{past}(\mathsf{D})^{\prec\overrightarrow{\mathsf{T}}}$ *be the past of* D*, then a policy,* $\delta_{\mathsf{D}}^{\prec\overrightarrow{\mathsf{T}}}$*, for* D *is the function:*

$$\delta_{\mathsf{D}}^{\prec\overrightarrow{\mathsf{T}}} : \mathbf{sp}(\mathbf{past}(\mathsf{D})^{\prec\overrightarrow{\mathsf{T}}}) \mapsto \mathbf{sp}(\mathsf{D}).$$

If $\mathbf{past}(\mathsf{D})^{\prec\overrightarrow{\mathsf{T}}}$ includes continuous variables an infinite number of policies exists for D. In order to deal with this, a policy of a decision, which has a continuous variable in its past, is taken based on a grouping of the states of continuous variables. We discuss this in detail later.
A strategy, Δ, of an IDIT is a set containing a policy for each decision in the IDIT, that is:

Definition 5.2 (Strategy, Δ)
Let $\mathbf{V}_{\mathsf{D}}$ *be a set of decisions, and each* D *in* $\mathbf{V}_{\mathsf{D}}$ *has a policy,* $\delta_{\mathsf{D}}^{\prec\overrightarrow{\mathsf{T}}}$*, then a strategy,* Δ *is:*

$$\Delta = \{\delta_{\mathsf{D}}^{\prec\overrightarrow{\mathsf{T}}} | \forall \mathsf{D} \in \mathbf{V}_{\mathsf{D}}\}.$$

We say a policy is an *optimal policy* if it maximizes the expected utility of the decision. A strategy containing only optimal policies is an *optimal strategy*, which we denote $\hat{\Delta}$.
The aim of a solution method is to find an optimal strategy for taking the decisions in an IDIT.

5.1.1 Outline of the Solution Method

The solution method for IDITs is structured on two levels. A global level, which decomposes the IDIT into symmetric IDITs, and merges these when a result is found, to get the solution for the IDIT. When the IDIT is decomposed to manageable pieces, each piece is solved using a local solution method.
The global level of the solution method is inspired by the solution methods for asymmetric decision problems, such as the ones in [Nielsen and Jensen, 2002], and [Demirer and Shenoy, 2001], and the local level is inspired by lazy evaluation, [Madsen and Jensen, 1999]. The idea is to decompose an IDIT, which includes asymmetries, into a number of symmetric IDITs. The symmetric IDITs are then solved individually and the results of the decompositions are merged to give the result of the original IDIT.

The outline of the parts of a solution method for IDITs, and what each part does, is as follows:

Splitting an IDIT: In order to find an elimination sequence for an IDIT the temporal ordering of the variables has to be found. As there is no unique temporal ordering for an IDIT, we split the IDIT into symmetric IDITs. The splitting is

done by finding the first variable imposing a split, based on a preliminary temporal ordering, which can be deduced from the original IDIT. By instantiating all variables, which lead to asymmetries, and splitting the IDIT according to these guards and restriction functions, all asymmetries are resolved in the resulting partially symmetric IDITs. This is continued until all asymmetries are resolved, thus a symmetric IDIT is constructed. This splitting of the IDIT is the first part of solving an IDIT. The splitting of an IDIT follows the approach described in Chapter 4.

Structure of eliminations: When a temporal ordering of the variables has been deduced, a method should structure the elimination of variables. Several approaches are applicable to do this. The goal of all approaches is to construct an elimination order from the temporal ordering, such that the elimination can be executed as efficiently as possible.

Elimination of variables: Having found an elimination order, the elimination commences. The elimination of variables is done for one node at a time, by following the principles of expected utility. In this manner an optimal policy for each decision can be found, and when all decisions have been eliminated the optimal strategy for the IDIT is found.

Merging of symmetric IDITs The splitting is done recursively, and whenever a symmetric IDIT is found, an elimination order for this symmetric IDIT is deduced. Then the variables before the last split variable, with respect to the elimination order, are eliminated, and optimal policies for the decision variables after this split are found. When the split variable is to be eliminated the results from each branch induced by the split variable is used to eliminate the split variable. That is, when an intermediate node in the split tree has received results from all its children, these results are merged, making the intermediate node a new leaf node. In this manner the optimal policies for all decisions are found.

The following sections elaborate further on each of the four steps presented above.

5.2 Splitting an Influence Diagram Involving Time

In Chapter 4 we described the temporal ordering of a symmetric IDIT, which is a way of resolving the asymmetries of the IDIT. This principle is also necessary in order to solve the IDITs as we need an order in which we can eliminate the variables to find an optimal strategy, [Jensen et al., 1994]. The effects of splitting an IDIT are three-fold in the solution method, as it divides the problem into sub-problems; it resolves the asymmetries within the IDIT, such that techniques inspired by those used for influence diagrams can be used; and a total ordering of decision and time variables emerges.

The strategy for splitting the IDIT follows the strategy for finding the temporal ordering. First, a preliminary temporal ordering is found using the qualitative level of the IDIT. Then a first time variable is found using the preliminary temporal ordering, and the set of splits this variable imposes is found. Then the IDIT is split on the time variable. By performing these operations in a recursive manner, we find a set of symmetric IDITs. The temporal ordering for each symmetric IDIT is established, so a structure of elimination can be constructed for each. By merging the result of each symmetric IDIT, resulting from a split, a solution for the IDIT is found. The specifics of how the structure of elimination and the merging is performed follows in the sections below.

Finding the Preliminary Temporal Ordering

To initiate the construction of the preliminary temporal ordering, the first decision in the IDIT must be identified. There is always one such decision, as proven in Chapter 4. It can be identified as the decision, for which the set of ancestors intersected with the set of decision variables is the empty set. That is, the first decision, D', of an IDIT is the decision for which it holds that $\mathbf{an}(D') \cap \mathbf{V}_D = \varnothing$.

As there cannot be any guards before the first decision the set of observed variables for this decision consists of only unguarded variables. The preliminary temporal ordering of an IDIT with a first decision, D, for which the set of observed variables is $\mathbf{I}_D$, therefore has a preliminary temporal ordering, $\mathbf{I}_D < D$.

If a decision is a decision involving time, the time variable associated with the decision is immediately after the decision variable in the preliminary temporal ordering, that is, if there exists a decision variable, D, and an associated time variable, T^e_D, then $D < T^e_D$.

Having identified the first decision, the set of observed variables for this decision, and, possibly, a time variable, all remaining variables should now be ordered in a similar manner. We order these variables by repeatedly identifying the next decision in the preliminary temporal ordering. As long as no decision, already placed in the preliminary temporal ordering, has a time variable associated with it, the next decision can be identified. The next decision is the one, for which the set of ancestors intersected with the set of decision variables equals all the previous decisions in the preliminary temporal ordering. That is, if the next decision is D_i and $\mathbf{P}$ is the set of decisions already ordered, then $\mathbf{an}(D_i) \cap \mathbf{V}_D = \mathbf{P}$.
Until the first time variable in the IDIT is placed in the preliminary temporal ordering, the decision variables before this time variable, and the observed variables for each of the decisions are ordered in this fashion. In the special case of IDITs, in which there does not exist a time variable, the temporal ordering yields an order similar to the one for influence diagrams as described in [Jensen et al., 1994].

Decisions, which are after the first time variable in the preliminary temporal ordering,

can have a set of observed variables, which differs according to the point in time they are taken. This is a result of guards on guarded arcs being evaluated to either true or false. The order of taking two or more decisions may also change due to different configurations of time variables.
Assuming that the part of an IDIT before the first time variable has been ordered according to the above specifications, and the decision variable following this time variable has been uniquely identified, then the set of observed variables for this decision can be found as previously, except for the chance variables, which have a guarded arc going to the decision. That is, we have an IDIT as the one illustrated in Figure 5.1.
Figure 5.1 depicts an IDIT in which: D_1 is the first decision involving time; $\mathbf{R}_0$ is the part of the IDIT before D_1; D_2 is the decision following immediately after D_1 in the preliminary temporal ordering; $\mathbf{I}_{\mathsf{D}_2} \cup \mathbf{I}^{\mathbf{g}}_{\mathbf{D}_2}$ is the set of chance variables between the two decisions, where $\mathbf{I}_{\mathsf{D}_2}$ is the set of chance variables always observed, and $\mathbf{I}^{\mathbf{g}}_{\mathbf{D}_2}$ is the set of guarded observed variables; and $\mathbf{R}_1$ is the set of variables after D_2, which may include a time variable associated with D_2, and $\mathbf{R}_1$ also includes $\mathbf{I}^{\mathbf{g}}_{\mathbf{D}_2}$, as defined previously. An arc emanating from a set of nodes going into a node in the figure represents that each element in the set has an arc going to that node, this is the case for the relation between $\mathbf{R}_0$ and D_1, whereas an arc from a node into a set of nodes represents a preliminary temporal precedence, such as the relation between D_2 and $\mathbf{R}_1$.

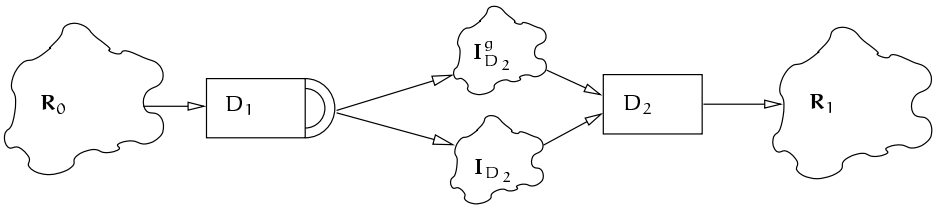


Figure 5.1: *Part of an IDIT.*

In the preliminary temporal ordering the two sets of observed variables, $\mathbf{I}_{\mathsf{D}_2}$ and $\mathbf{I}^{\mathbf{g}}_{\mathbf{D}_2}$, are before D_2, and the set of guarded observed variables, $\mathbf{I}^{\mathbf{g}}_{\mathsf{D}_2}$, is also in the set of chance variables never observed, or observed too late to have an impact on any decision, that is, we have a preliminary temporal ordering deduced as:

$$\ldots < \mathbf{I}_{\mathsf{D}_2} \cup \mathbf{I}^{\mathbf{g}}_{\mathsf{D}_2} < \mathsf{D}_2 < \ldots < \mathbf{I}^{\mathbf{g}}_{\mathsf{D}_2}$$

.

We can generalize this and say that, for any decision, D, in an IDIT, the preliminary temporal ordering, $<$, with respect to D, is deduced as $\mathbf{I}_{\mathsf{D}} \cup \mathbf{I}^{\mathbf{g}}_{\mathsf{D}} < \mathsf{D} < \mathsf{T}^{\mathsf{e}}_{\mathsf{D}} < \ldots < \mathbf{I}^{\mathbf{g}}_{\mathsf{D}}$, where $\mathbf{I}_{\mathsf{D}}$ is the set of observed variables, and $\mathbf{I}^{\mathbf{g}}_{\mathsf{D}}$ is the set of guarded observed variables, for D and $\mathsf{T}^{\mathsf{e}}_{\mathsf{D}}$ is the possible time variable associated with D.

Having specified the preliminary temporal ordering for any single decision, a possible associated time variable, and the set of observed variables for this decision, only the specification of how to find the ordering of multiple decisions after the first time variable remains.
From the definition of the qualitative part of an IDIT it is known that there is a path through all time and decision variables, this is specified in Rule (4) for the qualitative part of an IDIT. This path is used to find the next decision for some decision, assuming that all decisions before the current one is ordered in the preliminary temporal ordering. The decision read after the current decision, is the decision for which the set of ancestors is minimal, compared to the set of ancestors for all other decisions not yet considered. Using this definition it is known that, if the present decision, D, is not the last decision in the IDIT, then there exists a decision, D', which has D in its set of ancestors, as there must be a path between the two decisions according to the definition. If there are multiple decisions after the present one, the minimal set of ancestors is the set, which is a proper subset of all others. In cases where multiple cycles exist, such as was illustrated in Figure 4.9 the set of ancestors from unguarded arcs are used to order the decisions involved in the cycles.
Algorithm 5.1 specifies how a preliminary temporal ordering is found based on the qualitative level of an IDIT, $\mathcal{I}$.

PRELIMINARYTEMPORALORDERING($\mathcal{I} = (\mathbf{V}, \mathbf{L}, \mathbf{E})$)

1: Find decisions, $\mathbf{D}$, with minimal sets of ancestors.
2: **if** $\mathbf{D}$ contains one element, D **then**
3: Find the set of observed variables, $\mathbf{I}_{\mathsf{D}}$ and guarded observed variables, $\mathbf{I}^{\mathsf{g}}_{\mathsf{D}}$ for D
4: Insert $\mathbf{I}_{\mathsf{D}} \cup \mathbf{I}^{\mathsf{g}}_{\mathsf{D}} < \mathsf{D} < \mathsf{T}^{\mathsf{e}}_{\mathsf{D}}$ in PTO
5: Call PRELIMINARYTEMPORALORDERING($\mathcal{I}' = (\mathbf{V} \setminus \{\mathsf{D}\} \cup \mathbf{I}_{\mathsf{D}} \cup \{\mathsf{T}^{\mathsf{e}}_{\mathsf{D}}\}, \mathbf{L}, \mathbf{E})$
6: **else if** $\mathbf{D}$ is $\varnothing$ **then**
7: Insert $\mathbf{V}$ in PTO
8: **return** PTO
9: **else**
10: Find the set of observed variables, $\mathbf{I}_{\mathsf{D}}$ and guarded observed variables, $\mathbf{I}^{\mathsf{g}}_{\mathsf{D}}$ for each $\mathsf{D} \in \mathbf{D}$
11: Compare sets of ancestors from unguarded arcs for each pair, D, D'.
12: **if** $\mathbf{an}(\mathsf{D}) \subseteq \mathbf{an}(\mathsf{D}')$ **then**
13: Insert $\mathbf{I}_{\mathsf{D}} \cup \mathbf{I}^{\mathsf{g}}_{\mathsf{D}} < \mathsf{D} < \mathsf{T}^{\mathsf{e}}_{\mathsf{D}} < \mathbf{I}_{\mathsf{D}'} \cup \mathbf{I}^{\mathsf{g}}_{\mathsf{D}'} < \mathsf{D}' < \mathsf{T}^{\mathsf{e}}_{\mathsf{D}'}$ in PTO
14: **else**
15: Insert $(\mathbf{I}_{\mathsf{D}} \cup \mathbf{I}^{\mathsf{g}}_{\mathsf{D}} < \mathsf{D} < \mathsf{T}^{\mathsf{e}}_{\mathsf{D}}), (\mathbf{I}_{\mathsf{D}'} \cup \mathbf{I}^{\mathsf{g}}_{\mathsf{D}'} < \mathsf{D}' < \mathsf{T}^{\mathsf{e}}_{\mathsf{D}'})$ in PTO
16: Call PRELIMINARYTEMPORALORDERING($\mathcal{I}' = (\mathbf{V} \setminus \mathbf{D} \bigcup_{\mathsf{D} \in \mathbf{D}} \mathbf{I}_{\mathsf{D}} \cup \{\mathsf{T}^{\mathsf{e}}_{\mathsf{D}}\}, \mathbf{L}, \mathbf{E})$.

Algorithm 5.1: *The algorithm for constructing the preliminary temporal ordering of any IDIT, $\mathcal{I}$.*

5.2.1 Split Trees

The method for constructing the split tree for some IDIT is to first find the preliminary temporal ordering, $<$, of the variables. The method then constructs the root of the tree, which is the original IDIT. The first possible split variable can be found in the preliminary temporal ordering as the first time variable. The method splits on a time variable, if one or more decisions, which are before the next time variable in the preliminary temporal ordering, have a set of guarded observed variables, are not ordered in relations to other decisions, or have a restriction function.
For each guard or restriction the point in time associated with it is used to construct a minimal set of time intervals, such that no information is lost in an instantiation of the intervals. For instance, in Example 1, the SAR problem, when instantiating T^{e}_{Ws}, the guard between Hs and Se, which is true if $\mathsf{t} \geq 48$, and the restriction of the state space of *Search* are used to find the intervals: $[0;12[$, $[12;18[$, $[18;20[$, $[20;48[$, and $[48;168[$.
For each interval found in this manner, a new node is constructed and a branch connecting the current node to the new node is added. The partially symmetric IDIT in this node is an IDIT, in which the guards between the split variable and the next time variable are changed to true, if the guard evaluates to true, or removed if the guard evaluates to false. The evaluation of the guards is a result of the time variable being in the interval determined by the split. As the intervals of the split are constructed from the guards, no interval can exist in which it cannot be determined if the guard is true or false. Furthermore, the restriction function is evaluated and the result of this evaluation is set to be the state space of the decision.

The algorithm for constructing a split tree is given in Algorithm 5.2. Before the algorithm begins it is assumed that an initiation method has constructed the root of the split tree, found a preliminary temporal ordering, PTO, the first split variable, V, and the point in time or interval, T, setting the range for V in partially symmetric IDIT. The algorithm uses a set of list operations which should be self explanatory.

Algorithm 5.2 is compact with respect to elucidating comments. What happens is the following: we start in line 1 by identifying the next element in the preliminary temporal ordering, with respect to V. If this next element, or variable, is a decision variable with some restriction on it, either in form of a restriction function or a guarded arc, we take the point in time, or time interval, associated to that restriction or guard and store it in a list. Then we add the decision to a list of decisions, this is done in line 8. This we continue doing as long as we are not considering a time variable representing the end-time of some decision. Lines 10 and 11 state that, if we have considered the last variable in the IDIT, we simply return to the parent, with respect to the entire split tree. If we, at any time, consider a time variable, as just mentioned, or have considered the last variable in the IDIT, we order the time intervals we have gotten from the restriction functions and guards, according to each other. As long as the list of these time intervals is not empty we construct a child of the current partially symmetric IDIT. As seen in line 16, this child is a new split

```
SplitTree(V, I, PTO, T)
 1: V' ← PTO.NextElement(V)
 2: while V' ∉ V_T^e do
 3:    if V' ∈ V_D then
 4:       if γ_D ∈ Γ then
 5:          timeInterval ← timeInterval.AddElement(ExtractTime(V'))
 6:       for (X, V', l) ∈ E_g and l ≠ true do
 7:          timeInterval ← timeInterval.AddElement(ExtractTime(l))
 8:       decisions ← decisions.AddElement(V')
 9:    V' ← PTO.NextElement(V')
10:    if V' = null then
11:       return to parent
12: timeInterval ← timeInterval.Sort()
13: T_0 ← T
14: T_1 ← timeInterval.FirstElement()
15: repeat
16:    newnode ← SplitTree(V', SymmetricIDIT(I, T_0, decisions), PTO, T_0)
17:    AddEdge((thisnode, newnode, T_0 ≤ t < T_1))
18:    T_0 ← T_1
19:    T_1 ← timeInterval.NextElement(T_0)
20: until T_0 = null
21: return to parent
```

Algorithm 5.2: *The algorithm for constructing a split tree for any IDIT, $\mathcal{I}$.*

tree. In line 17 we add an edge between the new child and the node containing the partially symmetric IDIT we have been dealing with so far. This edge is labelled with whatever time interval we are building the new split tree from. Then, in line 19, we pick the next element in the list of time intervals, and this construction of children we repeat until no more time intervals exist in the aforementioned list.

Algorithm 5.3 implements SymmetricIDIT, which takes as input an IDIT, a point in time, for which the symmetric IDIT is constructed, and a list of decisions.

```
SymmetricIDIT(𝓘, T_0, decisions)
 1: for D ∈ decisions do
 2:   if γ_D ∈ Γ then
 3:     sp(D) ← {d_i : γ_{d_i}(T_0) = true}
 4:   for (X, D, l) ∈ E_g do
 5:     if l(T_0) = true then
 6:       E_g ← E_g.AddElement(X, D)
 7:       for (X, Y, l') do
 8:         E_g ← E_g.Remove((X, Y, l'))
 9:     else
10:       E_g ← E_g.Remove((X, D, l))
11:   return 𝓘
```

Algorithm 5.3: *The algorithm for constructing symmetric IDITs for any IDIT, $\mathcal{I}$.*

Algorithm 5.3 is also compact with respect to elucidating comments. What it does is to go through all decisions in the list of decisions it receives as input. Then, for all decisions, which are restricted, the state space is changed to comply with the restriction, this is seen in lines 2 and 3. And for all guarded arcs into each of these decisions, if the guard evaluates to true, as a consequence of the interval in which the current symmetric IDIT is, the arc is converted to an informational arc. If the guard evaluates to false, the arc is simply removed from the set of arcs. This happens in lines 4 through 10. In line 10 the resulting symmetric IDIT is returned. The symmetric IDIT is then put in the node representing the interval of T_0 in the split tree.

For the root and each internal node in the split tree a temporal ordering for the partially symmetric IDIT in it is constructed using Algorithm 5.1. This gives an ordering of the partially symmetric IDITs in these nodes, which is similar to the temporal ordering for the part of the partially symmetric IDIT before the time variable, which the partially symmetric IDIT splits on, and a preliminary temporal ordering of the rest.
For all leaf nodes in the split tree a method similar to Algorithm 5.1 is used. However, the sets of guarded variables are not considered and the part, which handles multiple decisions, which cannot be ordered in relation to each other, is removed too.

5.3 Structure of Elimination

After splitting the original IDIT into partially symmetric and symmetric IDITs, we use an approach inspired by a solution method for influence diagrams, to solve the IDITs. We only solve the leaves of a split tree, meaning that an internal node receives a result of elimination from each of its children, before the local solution of the internal node begins, eliminating the leaf nodes in the process. The part of the partially symmetric IDIT, which is to be solved is therefore the part between the split variable inducing this partially symmetric IDIT and the split variable, which splits it into its children. This part of the partially symmetric IDIT is symmetric, thus, the local solution method of internal nodes is no different than the local solution method for leaf nodes, except for the merging of its children. A solution of a symmetric IDIT is to find an optimal policy for all decision variables, which are before the variable that caused the split, in the elimination order. In cases where there are no decision variables the result of eliminating all variables before the split variable in the elimination order is passed on to the parent node in the split tree. If no more split variables exist, the rest of the variables are eliminated in the same fashion, still respecting the elimination order.

We have chosen to structure the elimination of variables using *strong junction trees*, [Jensen et al., 1994], as this method of structuring the elimination has several benefits: a strong junction tree is a different representation of a DPIT than a symmetric IDIT, thus the conversion to a strong junction tree liberates the solution method of respecting the rules imposed by the definition of IDITs; furthermore, strong junction trees are efficient for retrieving an optimal strategy for a decision problem, [Jensen et al., 1994]; and it directly depicts the conditional independence of the symmetric IDIT.
A different approach of structuring the order of elimination, is to use the ideas of *node-removal and arc-reversal* as proposed by [Shachter, 1986], which is also efficient. An approach based on this method works directly on the symmetric IDIT. Thus, it does not have the benefits achieved through strong junction trees.

An interesting aspect, which should be noted, is that the deduction of the temporal ordering described above, and the elimination of variables, which is described in Section 5.4, enables future solution methods to use, for instance, node-removal and arc-reversal instead of strong junction trees, without changing these parts of the method.

5.3.1 Moralizing Influence Diagrams Involving Time

In order to construct a strong junction tree the symmetric IDITs have to be structured into cliques, according to the elimination order, using strong triangulation. We triangulate the symmetric IDIT by first converting it to a moral graph, which is an undirected graph, in which all immoralities have been removed. In this section we

describe how a symmetric IDIT is moralized.

A moralization of a directed graph, $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, is the undirected graph, $(\mathbf{V}', \mathbf{E}')$ resulting from removing all immoralities. It should be noticed that the set of labels have been omitted as the set of labels of a symmetric IDIT consists only of the label, true. The idea is to first remove all informational arcs from the IDIT. Then remove all immoralities, by adding an edge between nodes sharing a child, if this edge does not already exist, and finally, remove all utility nodes.
Algorithm 5.4 specifies what is meant by moralization in the thesis.

MORALIZE($\mathcal{I}$)

1: Remove all informational arcs
2: **for** each node, V, in $\mathcal{I}$ **do**
3: Add edge between each pair of parents of V, if it does not already exist
4: Remove all utility nodes
5: Undirect the graph
6: **return** resulting graph

Algorithm 5.4: *An algorithm for constructing the moral graph of any IDIT, $\mathcal{I}$.*

5.3.2 Strong Triangulation

Before constructing a strong junction tree, we triangulate the graph. The idea of this triangulation is to ensure that when a node is to be eliminated all its neighbours are connected. This is ensured by going through the elimination order and for each node adding fill-ins between neighbours, which are not connected, and which have not already been considered. When we have completed adding fill-ins the resulting graph is triangulated. We eliminate the variables in an elimination order, which respects the reverse of the temporal order.
As there can be many elimination orders, due to the chance variables not being ordered, and different orders yield different strong triangulations, we strive to find the triangulation which is minimum. A triangulation can be minimum in different ways. For instance, a minimum triangulation can be the triangulation, which adds the least amount of fill-ins, or the one for which the sum of the clique sizes is minimum. We refer to [Kjærulff, 1993] for a discussion of minimum triangulations. No matter what approach of minimum triangulation is chosen, finding it is NP-hard [Jensen and Jensen, 1994]. This means that, in order to complete it in a reasonable amount of time, some heuristics have to be applied. We have chosen to use minimum fill-in triangulation. To this we apply a heuristic in the form of having a look-ahead of two, so if two or more chance variables are not ordered we examine all possible combinations of these variables. We add the fill-ins necessary for the triangulation of the graph when eliminating the first two variables, and choose the triangulation which adds the fewest fill-ins. If two combinations both have the least

amount of fill-ins we eliminate the next variable to see if there is a difference, and if not, we choose one of these at random. Other alternatives, include using minimal separator sets, [HUGIN Expert, 2003], which seems to work quite well.

The elimination order orders the variables in a total ordering. We define a function, α, which maps each node in the elimination ordering to the natural number according to when it is eliminated. We define α to be the bijection, $\alpha : \mathbf{V} \leftrightarrow \{1, \ldots |\mathbf{V}|\}$, where $\mathbf{V}$ is the set of nodes in the elimination order, such that, if V is before V', according to the elimination order, then $\alpha(V) < \alpha(V')$.

The algorithm for triangulation takes as its arguments an undirected graph and is structured as presented in Algorithm 5.5.

TRIANGULATION($\mathcal{M} = (\mathbf{V}, \mathbf{E})$)

1: $\mathbf{V}' \leftarrow \mathbf{V}$
2: **for** $\alpha(V) = 1$ to $|\mathbf{V}|$ **do**
3: **for** each $X, Y \in \mathbf{ne}(V)$ **do**
4: $\mathbf{E} \leftarrow \mathbf{E} \cup \{\{X, Y\}\}$
5: $\mathbf{V}' \leftarrow \mathbf{V}' \setminus \{V\}$
6: **return** ($\mathbf{V}$, $\mathbf{E}$)

Algorithm 5.5: *An algorithm for setting up a strong triangulation of any moralized graph, $\mathcal{M}$.*

The result of Algorithm 5.5 is the triangulated graph from which the strong junction tree is constructed.

5.3.3 Strong Junction Tree

A strong junction tree is a rooted tree of cliques, which is constructed such that elimination of variables can be performed using an absorption method.

The graph resulting from the strong triangulation can be divided into a set of cliques, $\mathbf{K}$, by following the elimination order. These cliques are organized in a strong junction tree, $\mathcal{T}$, for which it holds, that:

- For each pair of cliques, $\mathbf{C}$ and $\mathbf{C}'$, in $\mathcal{T}$, the set $\mathbf{C} \cap \mathbf{C}'$ is in all cliques on the path between $\mathbf{C}$ and $\mathbf{C}'$.
- For each pair of adjacent cliques, $\mathbf{C}$ and $\mathbf{C}'$, in $\mathcal{T}$, the intersection, $\mathbf{C} \cap \mathbf{C}'$, is associated as a *separator* between the two.
- There exists a strong root, and for each pair of adjacent cliques, $\mathbf{C}$ and $\mathbf{C}'$, in $\mathcal{T}$, where $\mathbf{C}$ is closest to the strong root, the variables of the set, $\mathbf{C} \cap \mathbf{C}'$, are after the variables of the set $\mathbf{C}' \setminus \mathbf{C}$ in the elimination order.

In a strong junction tree the elimination order is structured such that the solution is found by marginalizing free variables, and propagating the resulting potentials towards the root. The construction of the strong junction tree is described in Algorithm 5.6 which takes a triangulated graph as its argument.

StrongJunctionTree($\mathcal{G} = (\mathbf{V}, \mathbf{E})$)

1: $\mathbf{i} \leftarrow 1$
2: **while** $\mathbf{i} \leq |\mathbf{V}|$ **do** (Find candidates to cliques.)
3: $\mathbf{C_i}$ is set to be the clique containing a variable V, where $\mathbf{i}$ equal $\alpha(V)$.
4: $\mathbf{K} \leftarrow \mathbf{K} \cup \mathbf{C_i}$
5: **for** $\mathbf{C_i}, \mathbf{C_j} \in \mathbf{K}$ **do** (Removes cliques which are subsets of another clique)
6: **if** $\mathbf{C_i} \subseteq \mathbf{C_j}$ **then**
7: $\mathbf{K} \leftarrow \mathbf{K} \setminus \mathbf{C_i}$
8: $\mathbf{S} \leftarrow \mathbf{E}' \leftarrow \varnothing$
9: $\mathbf{K}' \leftarrow \{\mathbf{C_j}\}$ ($\mathbf{C_j}$ is the clique with lowest index in $\mathbf{K}$.)
10: **while** $\mathbf{K}' \neq \mathbf{K}$ **do** (Constructing the tree.)
11: **pick** $\mathbf{C_i} \in \mathbf{K} \setminus \mathbf{K}'$ **s.t.** $\exists \mathbf{C_k} \in \mathbf{K}' | \mathbf{C_i} \cap \mathbf{C_k} \neq \varnothing$
12: $\mathbf{S} \leftarrow \mathbf{S} \cup \{\mathbf{C_i} \cap \mathbf{C_k}\}$
13: $\mathbf{E}' \leftarrow \mathbf{E}' \cup \{\{\mathbf{C_i}, \mathbf{C_k}, \mathbf{C_i} \cap \mathbf{C_k}\}\}$
14: $\mathbf{K}' \leftarrow \mathbf{K}' \cup \{\mathbf{C_i}\}$
15: **return** $(\mathbf{K}, \mathbf{S}, \mathbf{E}')$

Algorithm 5.6: *An algorithm for constructing a strong junction tree, of a triangulated graph, $\mathcal{G}$.*

Algorithm 5.6 is similar to the algorithm of [Jensen et al., 1994], in which the correctness of the algorithm is argued for.

Following the construction of a strong junction tree we associate to each clique in the strong junction tree two sets of potentials, which reflect the quantitative level of the symmetric IDIT.
Generally, a probability distribution, $\phi = \mathsf{P}(\mathbf{X}|\mathbf{Y})$, can be called a probability potential. A probability potential is a function, ϕ, which maps the state space of a set of variables, $\boldsymbol{W} = \mathbf{X} \cup \mathbf{Y}$, into a positive real number, that is, $\phi : \mathbf{sp}(\boldsymbol{W}) \mapsto \mathbb{R}^+$. The set of variables, $\boldsymbol{W}$, is called the domain of ϕ and is denoted as $\mathbf{dom}(\phi)$. Two probability potentials can be multiplied to find the potential for the joint distribution. Other properties of potentials are described in [Jensen, 2001]. Furthermore, we specify division of two potentials to be the same as when two reals are divided, with the exception that, if the denominator is zero we define the result to be zero as well. In a similar manner as done for probability distributions, utility functions can be viewed as utility potentials. It should be noted that this can impose a positive linear transformation of the utility function to satisfy the function mapping to a positive real. In accordance to the decomposition of the utility function as local utility functions, two utility potentials may be summed.

To each clique, $\mathbf{C}_i$, in the strong junction tree, we associate two sets of potentials, $\mathbf{\Phi}_{\mathbf{C}_i}$ and $\mathbf{\Psi}_{\mathbf{C}_i}$. $\mathbf{\Phi}_{\mathbf{C}_i}$ is the set consisting of all probability potentials, for which $\mathbf{dom}(\phi) \subseteq \mathbf{C}_i$, that is, $\mathbf{\Phi}_{\mathbf{C}_i} = \{\phi \in \mathbf{\Phi} \cup \mathbf{\Pi} | V \in \mathbf{dom}(\phi), V \in \mathbf{C}_i\}$, and $\mathbf{\Psi}_{\mathbf{C}_i}$ is the set consisting of all utility potentials, for which $\mathbf{dom}(\psi) \subseteq \mathbf{C}_i$, that is, $\mathbf{\Psi}_{\mathbf{C}_i} = \{\psi \in \mathbf{\Psi} | V \in \mathbf{dom}(\psi), V \in \mathbf{C}_i\}$.

5.4 Elimination of Variables

Having constructed a strong junction tree the solution method proceeds by eliminating the free variables in the symmetric IDIT in accordance to the strong junction tree. The manner in which a variable is eliminated depends on its type and whether or not it has a continuous variable in its set of parents. In this section we describe how each type of variable is eliminated. Finally, we describe how the elimination is carried out in strong junction trees.

A node, $\mathbf{C}_i$, in a strong junction tree represents a clique of variables, $\{V_1, V_2, \ldots, V_n\}$, and has the two sets of potentials, $\mathbf{\Phi}_{\mathbf{C}_i}$ and $\mathbf{\Psi}_{\mathbf{C}_i}$, associated with it. Eliminating a variable, V_i, from $\mathbf{C}_i$ is done by marginalizing V_i from all probability potentials, ϕ, where V_i is in the domain of ϕ in $\mathbf{\Phi}_{\mathbf{C}_i}$, and remove it from all utility potentials, ψ in $\mathbf{\Psi}_{\mathbf{C}_i}$, and then updating the two sets of potentials accordingly.
Let $\mathbf{\Phi}_{V_i}$ be the set of probability potentials, which have V_i in their domain and $\mathbf{\Psi}_{V_i}$ the set of utility potentials, which have V_i in their domain; furthermore, let $\mathbf{\Phi}^*_{V_i}$ be the set of probability potentials resulting from marginalizing V_i from $\mathbf{\Phi}_{V_i}$ and $\mathbf{\Psi}^*_{V_i}$ the set of utility potentials resulting from marginalizing V_i from $\mathbf{\Psi}_{V_i}$; and finally let $\bigmarg_{V_i}$ represent the manner in which V_i is marginalized. That is, generally, if V_i is a chance variable $\bigmarg_{V_i}$ represents summation, and if V_i is a decision variable $\bigmarg_{V_i}$ represents maximization, we elaborate on this below. Elimination of V_i is then:

$$\mathbf{\Phi}^*_{V_i} = \bigmarg_{V_i} \prod_{\phi \in \mathbf{\Phi}_{V_i}} \phi, \text{ and} \tag{5.1}$$

$$\mathbf{\Psi}^*_{V_i} = \bigmarg_{V_i} \prod_{\phi \in \mathbf{\Phi}_{V_i}} \phi \cdot \sum_{\psi \in \mathbf{\Psi}_{V_i}} \psi. \tag{5.2}$$

After finding $\mathbf{\Phi}^*_{V_i}$ and $\mathbf{\Psi}^*_{V_i}$, the two sets of potentials are updated to:

$$\mathbf{\Phi}^* = (\mathbf{\Phi} \cup \mathbf{\Phi}^*_{V_i}) \setminus \mathbf{\Phi}_{V_i} \text{ and} \tag{5.3}$$

$$\mathbf{\Psi}^* = \left(\mathbf{\Psi} \cup \frac{\mathbf{\Psi}^*_{V_i}}{\mathbf{\Phi}^*_{V_i}}\right) \setminus \mathbf{\Psi}_{V_i}. \tag{5.4}$$

The division of $\mathbf{\Psi}^*_{V_i}$ by $\mathbf{\Phi}^*_{V_i}$ is done in order to compensate for the multiplication in Equation 5.2. The method presented above shows a general manner of eliminating variables, however, the exact marginalization has not been specified. In the following sections we specify what $\bigmarg_{V_i}$ means for the different types of variables.

5.4.1 Elimination of Discrete Chance Variables

Discrete chance variables, which are not dependent on a time variable, are eliminated by summation over the chance variable being marginalized out. That is, in Equations 5.1 and 5.2, the universal marginalization operator is the sum over V_i.

Marginalization of chance variables dependent on time is conceptually equivalent to marginalization of chance variables, which are not dependent on time.
However, even though the two marginalizations are conceptually similar, the outcomes of the two are not. Whenever we marginalize a chance variable, which is influenced by time, we need to deal with functions. We divide the cases in the ones where a discrete chance variable, C, is influenced by time and those where it influences time.
Let C be the discrete chance variable, which is about to be eliminated. The product of the probability potentials including C in their domain is ϕ, where $\mathbf{dom}(\phi) = \{X\} \cup \{C\} \cup \mathbf{A}$, where $\mathbf{A}$ is a set of discrete variables and X is a time variable. Marginalizing C in this case results in a new potential where each configuration of $\mathbf{A}$ is associated to a function. The function is found as the sum, over the states of C, of the functions of this configuration of the original probability potential. If the potential includes two continuous variables, according to Chapter 4, we know that $P(T|T')$ is $P(T) + t'$. We utilize this to obtain a function as described above only over both time variables.

When C influences a time variable, and is eliminated before the time variable the product of probability potentials includes a joint over these two variables. When marginalizing C the result is a probability potential over the time variable. This potential includes a new function, instead of the density function of the time variable, for each configuration of discrete variables in the potential. This function is the sum of the density functions found by the state of C given this configuration of the discrete variables.

Marginalizing from the utility potential is done in a similar manner, but where the functions are multiplied by the function of the utility given the configuration of the discrete variables.

We have chosen to restrict the models of IDITs, to not allow one variable to be in the conditioning set of multiple time variables.

5.4.2 Elimination of Discrete Decision Variables

The elimination of discrete decision variables is done differently than the elimination of discrete chance variables. Decisions are marginalized from $\mathbf{\Phi}$ and $\mathbf{\Psi}$ using maximization, as opposed to summation. Decisions are marginalized from $\mathbf{\Phi}$ by choosing any choice, yielding a new potential from which the decision has been marginalized. Decision variables are eliminated from utility potentials by maximizing the outcome

given observations and previous decisions, according to an assumption that all future decision variables have been taken following this principle too.
When decisions do not have any continuous variables as parents, this maximization can be found as in influence diagrams, by taking the choice yielding the best result in the utility potential. That is, in this case 5.2 becomes:

$$\mathbf{\Psi}^*_{\mathrm{D}} = \arg\max_{\mathrm{D}} \prod_{\phi \in \mathbf{\Phi}_{\mathrm{D}}} \phi \sum_{\psi \in \mathbf{\Psi}_{\mathrm{D}}} \psi$$

Equation 5.1, for decision variables, is found by choosing any choice of D, as the resulting potentials are equivalent.
Having a time variable in the conditioning set does not change the calculation of $\mathbf{\Phi}^*_{\mathrm{D}}$, however the outcome, is changed as the resulting potential has a function over the time for each configuration of the variables in its domain.
If the decision we are marginalizing has a time variable as a parent, we get a utility potential with a continuous function for each state of the decision. By finding the intersections of these functions, we find the intervals where one choice is better than another, and this way we find the optimal strategy for each such interval. An example of this is seen in Figure 5.2.

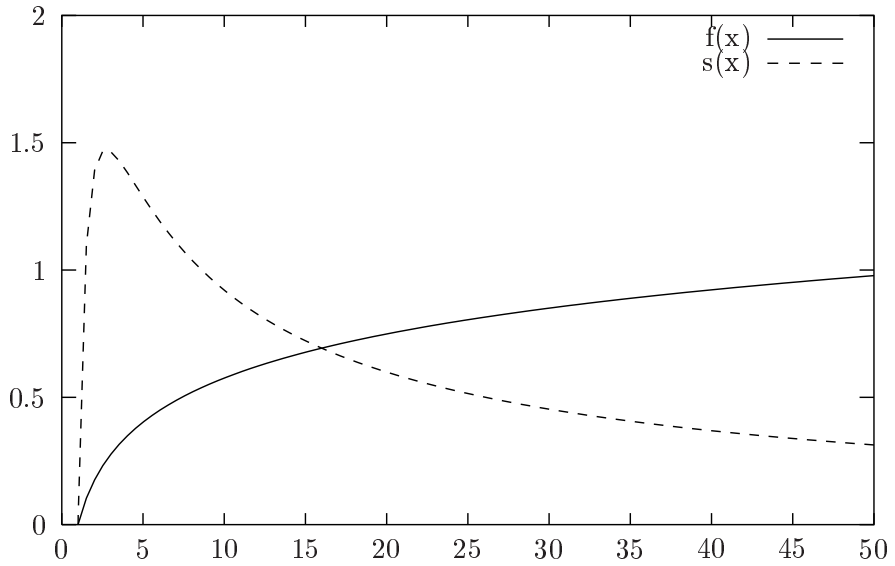


Figure 5.2: *Two utility functions representing the development over time given either the choice of* d_1, $\mathrm{f}(\mathrm{x})$, *or* d_2, $\mathrm{s}(\mathrm{x})$, *of* D.

When computing this, there can be an infinite number of such intervals if one of the functions is periodic or fluctuates. We restrict IDITs from using such utility functions.

5.4.3 Elimination of Time Variables

Elimination of time variables is done by integration over the entire state space of the variables. When the time variable is a split variable the integration is done over the interval of each symmetric IDIT, and the result is the sum of all integrals. For simplicity we say that if the time variables is not a split variable it has the interval $[0;\infty[$, and we generalize the updating of potentials, so Equations 5.1 and 5.2 become:

$$\Phi_T^* = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \phi_i dt \text{ and}$$

$$\Psi_T^* = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \phi_i \cdot \psi_i dt,$$

where ϕ_i and ψ_i denotes the probability and utility potentials for each interval. This is the ideal elimination. Unfortunately we cannot do this, as we have chosen to represent time variables by χ^2-distributions, which are not on closed form, meaning we cannot perform an exact integration over the state space of our time variables. This means that we have to use other methods to aid us. Such methods could be discretization of the time variables, or using some sampling technique. We have chosen the latter. We have chosen sampling, as we feel a discretization would result in losing too much expressive power with respect to representing time. One could argue that the discretization could be performed using any granularity, but the information of the intervals must be stored somewhere, so our choice is a matter of space. The exact sampling technique and algorithms for it are presented in Section 5.7, here we only show the ideal marginalization.

Time variables, which have a continuous variable in the conditioning set are marginalized in a similar manner. The difference is that the intervals go from t_i+t' to $t_{i+1}+t'$, where t' is the outcome of the time variable in the conditioning set of T.

5.4.4 Elimination of Wait Decision Variables

An optimal choice of a wait decision has similar properties as the optimal choice of a discrete decision variable, however, as wait decisions have a continuous state space, simply taking the maximum argument is not possible. One idea could be to discretize the wait decision. The reason for not doing this is similar to that for not discretizing time variables.
Assuming that the product of probability potentials including a wait decision multiplied by the sum of utility potentials including the same wait decision is a differentiable function, it is possible to find the choice of the wait decision, which results in the optimal strategy for the decision. This is done by examining the function for extrema, and selecting the extremum yielding the maximum value. By finding for which values of the range the derivative of the function equals zero, and comparing

the domain value of each of these values with each other the global extremum is found. The derivative can be found using a gradient descent method. This can only be done if the wait decision has no time variable as a parent.
In other words, the ideal marginalization of a wait decision is equal to the elimination of discrete decision variables, where $\underset{D}{\mathtt{argmax}}$, in the case of wait decisions, means to find the maximum value of a continuous function.
In the case where the wait decision we want to eliminate has a time variable as a parent, we find the optimal strategy in the same manner, but when evaluating the result, we set it to $t - t'$, where t is the interval or point in time found to result in the optimal strategy, and t' is the point in time represented by the time variable, which is a parent of the wait decision.
Updating $\mathbf{\Phi}$ when eliminating a wait decision is done in the same manner as when eliminating a discrete decision variable.

5.4.5 Message Passing and Marginalization

In this section we primarily focus on a solution in which it is assumed that we can find the exact probability distribution for our density functions, that is, calculate their integrals, and then later we show how an approximated value can be found.

We have now described how each type of variables is eliminated. Returning to the strong junction tree we can use this to find the optimal strategy for an IDIT.

It is assumed that a strong junction tree, $\mathcal{T}$, has been constructed from the symmetric IDIT, which we are solving. In $\mathcal{T}$ there are two adjacent cliques, $\mathbf{C}_i$ and $\mathbf{C}_j$, and they are separated by the separator set, $\mathbf{S}$. To $\mathbf{C}_i$ there are two sets associated, $\mathbf{\Phi}_{\mathbf{C}_i}$ and $\mathbf{\Psi}_{\mathbf{C}_i}$. There are two similar sets associated to $\mathbf{C}_j$. Furthermore, $\mathbf{C}_i$ is closer to the root of $\mathcal{T}$ than $\mathbf{C}_j$.

Lazy propagation, which is the approach our solution method is inspired by, uses message passing between cliques, and propagates these messages from the leaves to the root of $\mathcal{T}$. The messages are collected to the root by recursively invoking a message request from all underlying cliques, that is, cliques further from the root, and adjacent to the current one. In our example this means that $\mathbf{C}_i$ invokes a message request in $\mathbf{C}_j$ and all other underlying cliques, and when the underlying cliques have computed a message, each passes its message back to $\mathbf{C}_i$. When $\mathbf{C}_i$ receives these messages they are absorbed and then passed along as a single message to the overlying node. That is, a clique adjacent to $\mathbf{C}_i$, and closer to the root.

Absorption of messages from one clique, $\mathbf{C}_j$, into another, $\mathbf{C}_i$, which are separated with $\mathbf{S}$, means to marginalize the variables of $\mathbf{C}_j \setminus \mathbf{S}$ from $\mathbf{\Phi}_{\mathbf{C}_j}$ and $\mathbf{\Psi}_{\mathbf{C}_j}$ and from all the sets $\mathbf{\Phi}_{\mathbf{S}'}$ and $\mathbf{\Psi}_{\mathbf{S}'}$, where $\mathbf{S}'$ is a separator set of an underlying clique adjacent to $\mathbf{C}_j$ in $\mathcal{T}$. The result of marginalizing the variables is two sets of potentials, $\mathbf{\Phi}_{\mathbf{S}}$ and $\mathbf{\Psi}_{\mathbf{S}}$, which are associated to $\mathbf{S}$ as the result of absorbing $\mathbf{C}_j$ in $\mathbf{C}_i$. These sets are used as $\mathbf{C}_i$ passes its message further up the tree. In Algorithm 5.7 we present

how absorption of potentials is done for IDITs.

ABSORPTION

1: $\mathbf{R}_{\mathbf{S}} \leftarrow \mathbf{\Phi}_{\mathbf{C}_j} \cup \mathbf{\Psi}_{\mathbf{C}_j} \cup \bigcup_{\mathbf{S}' \in \mathbf{ch}(\mathbf{C}_j)} \mathbf{\Phi}_{\mathbf{S}'} \cup \mathbf{\Psi}_{\mathbf{S}'}$
2: Marginalize all variables not in $\mathbf{S}$ from $\mathbf{R}_{\mathbf{S}}$.
3: Associate $\mathbf{\Phi}_{\mathbf{S}}$ and $\mathbf{\Psi}_{\mathbf{S}}$ with $\mathbf{S}$ as the result of absorbing $\mathbf{C}_j$ in $\mathbf{C}_i$.

Algorithm 5.7: *The algorithm for absorption of potentials in IDITs.*

Algorithm 5.7 does not specify how variables are marginalized from the potentials. The marginalization is done in accordance to the type of variables being eliminated, using the rules described previously. Generally, however, the algorithm is as in Algorithm 5.8. We assume the algorithm returns the optimal choice when eliminating a decision variable.

MARGINALIZATION

1: Construct two sets, $\mathbf{\Phi}_V$ and $\mathbf{\Psi}_V$, which contain every ϕ and ψ, respectively, in any of the $\mathbf{\Phi}$, and $\mathbf{\Psi}$, where V is in the domain of either of ϕ or ψ.
2: Calculate $\mathbf{\Phi}^*_V$ and $\mathbf{\Psi}^*_V$
3: return $\mathbf{\Phi}^* = (\mathbf{\Phi} \cup \mathbf{\Phi}^*_V) \setminus \mathbf{\Phi}_V$ and $\mathbf{\Psi}^* = (\mathbf{\Psi} \cup \frac{\mathbf{\Psi}^*_V}{\mathbf{\Phi}^*_V}) \setminus \mathbf{\Psi}_V$.

Algorithm 5.8: *The general algorithm for marginalization of variables in IDITs.*

We do not give a formal proof for the algorithm, but refer to [Madsen and Jensen, 1999] for the proof when dealing with influence diagrams, and based on this we argue that the introduction of a set of continuous variables does not alter this result. As continuous chance variables are essentially chance variables with an infinite state space, the difference when marginalizing these as opposed to discrete chance variables is how to sum over the state space. However, neither ABSORPTION nor MARGINALIZATION specifies the marginalization operator, which is determined by the type of node being marginalized. The additions to MARGINALIZATION is to include the continuous variables when finding the utility potential, in a similar manner as for discrete variables, and to update the set of continuous probability potentials. This goes for both steps two and three in the algorithm.

The use of continuous chance and decision variables in IDITs yields it necessary to use integration and differentiation when solving an IDIT. As these integrations cannot be done in an exact manner, some approximation method has to be used. In [Broe et al., 2003] the solution sketch uses a numerical approximation, however, neither the appropriateness nor alternatives of this approximation were discussed. In Section 5.7 we discuss how we find the approximated values.

5.5 Merging of Symmetric Influence Diagrams Involving Time

In Section 5.2 we described how the original IDIT is split into partially symmetric and symmetric IDITs, and in Sections 5.3 and 5.4 we showed how each of these partially symmetric and symmetric IDITs are solved individually. In this section we describe how the solutions of two symmetric IDITs are merged. The symmetric IDITs are merged on two levels. First the strong junction trees from the symmetric IDITs are merged, and then the sets of potentials are merged.
In this section we describe the merging of two symmetric IDITs, as merging more than two is done by first merging two, and then viewing this merger as one symmetric IDIT. This is then merged to another and so on until no more symmetric IDITs need merging.

5.5.1 Merging Strong Junction Trees

From each of the subtrees resulting from splitting on a split variable, V, there exists a strong junction tree, $\mathcal{J}$. These strong junction trees have an equivalent subtree, as they are all constructed from symmetric IDITs based on the same partially symmetric IDIT. If any two of these strong junction trees should not have this property, the adding or removal of an arc in the part of the symmetric IDIT following the split node, should impose an additional arc in the part preceding the split node. We can, thus, reduce the problem to whether the removal of an arc, due to different symmetric IDITs, changes the structure of the part of the strong junction tree constructed from the part of the symmetric IDIT, which is before the split variable in the temporal ordering. If this part of the tree differs because of different symmetric IDITs, then the merging of two strong junction trees, is impossible, yielding the strong junction tree method unusable for solving symmetric IDITs.
Theorem 5.3 secures that the two *sub-junction trees* are equivalent.

Theorem 5.3 (Sub-Junction Tree Equivalence)
If $\mathcal{I} = (\mathsf{V}, \mathsf{L}, \mathsf{E})$ is an IDIT, T a time variable, D a decision variable, and C a chance variable in V, $(\mathsf{C}, \mathsf{D}, \mathsf{g})$ a guarded arc in E_{g}, where g references T, $\mathcal{I}'$ and $\mathcal{I}''$ two symmetric IDITs resulting from a split on T, such that g can be evaluated, and $\mathcal{T}'$ and $\mathcal{T}''$ the two strong junction trees constructed from $\mathcal{I}'$ and $\mathcal{I}''$, respectively. Then the sub-junction trees of $\mathcal{T}'$ and $\mathcal{T}''$, from the root to the clique, from which T is eliminated, are equivalent.

Proof: We prove Theorem 5.3 by arguing that no matter the elimination order of two variables, C and D, the cliques constructed by a strong triangulation for all variables after C and D are unchanged. When the cliques are unchanged, then the part of the strong junction tree, which is constructed from this set of cliques, is equivalent no matter the future.

If a clique closer to the strong root should be affected, one of the two elimination orders must add an arc between the two nodes, X and Y, in the clique. An edge is added during the strong triangulation if there is a path between X and Y, and all other nodes on this path are eliminated before them. However, as the moralization of two symmetric IDITs is similar even though they impose different elimination orders, due to the arc, (C, D, g), being an informational arc, the path is either in both symmetric IDITs or not at all, as both elimination orders have C and D before X and Y. Thus, such an arc cannot be added.

■

This problem is similar to that of finding the value of information for some chance variable. The solution to finding the value of information in strong junction trees is proposed in [Dittmer and Jensen, 1997].

The merging of two strong junction trees, therefore, reduces to picking one of the strong junction trees from the children and using this as the strong junction tree for the partially symmetric IDIT. As there always is a first interval we pick the strong junction tree resulting from the symmetric IDIT of this interval.

5.5.2 Merging Potentials

Theorem 5.3 ensures that the structure of two strong junction trees can be merged, it does, however, not ensure that the associated potentials are equivalent. Two potentials can differ by either not having the same domain, or having different values for similar configurations. In this section we discuss how to merge the potentials from two different symmetric IDITs.
If two probability potentials have different values, either some calculation has gone wrong, or the use of approximation has proposed two different values. As the algorithm is unambiguous, the first case cannot occur. In the second case, two approximations yield two different results, as these are both approximations, none of them are the exact result, and we have no way of finding which is the better approximation, in such cases we therefore choose one of the two to be the correct probability potential.
In cases where the domain of two probability potentials differ, for instance, $P(\mathbf{X}|\mathbf{Y})$ and $P(\mathbf{X}|\mathbf{Y},\mathbf{Z})$, we claim that the set of variables, $\mathbf{X}$ are conditionally independent of $\mathbf{Z}$ given $\mathbf{Y}$. The claim can be proven using the fact that the difference in probability potentials occurs due to the addition of edges during the strong triangulation. We know from Chapter 4 that the absence of an arc between two variables implies that the variables are conditionally independent given the past, thus, if $\mathbf{X}$ should be dependent on $\mathbf{Z}$, there must be a connection between the variables in the moralized graph.

The actual merging is done by comparing the potentials for each variable in the clique. If there are two potentials, $P(V|\mathbf{X})$ and $P(V|\mathbf{X}')$, then the combined potential is $P(V|\mathbf{X}' \cap \mathbf{X})$. We merge utility potentials by associating the utility potential achieved

by the split into some interval $[t_0; t_1[$, to the states of the time variable, which are within this interval.

5.6 The Solution Method

In the previous sections we have described how each step of the solution method is executed, yielding a local result. In this section we propose how the overall solution method is executed by use of Algorithms 5.1 to 5.8. We do not propose an explicit algorithm, as this would be a matter of fitting all previous algorithms together, by adding how each algorithm calls the next, and what is to be returned after an execution. We leave this for future implementations, and concentrate on the idea of the overall solution method. Figure 5.3 illustrates how the solution method should divide the problem using a global solution method, and describe how each subproblem should be solved using a local solution method. The numbers in the figure are used as both the order of the actual execution and as a reference point for a more thorough description, following the figure.

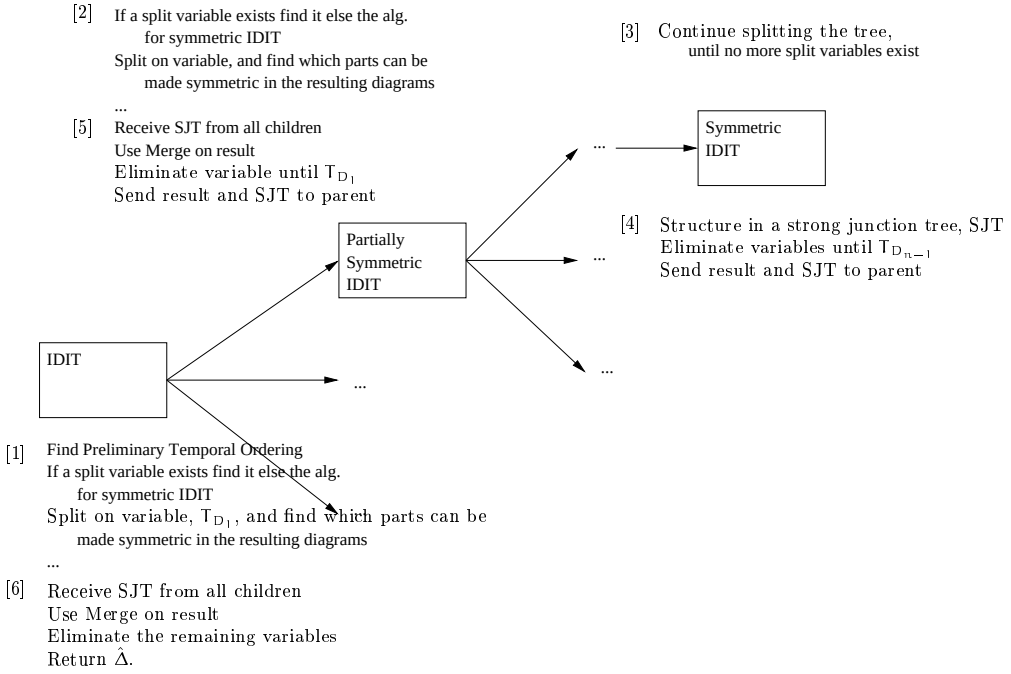


Figure 5.3: *An overview of the entire solution method.*

Figure 5.3 illustrates the elements of the overall solution method. Initially, an IDIT is given to the method as an argument. In [1] a preliminary temporal ordering, using Algorithm 5.1, is found. This establishes a temporal ordering until the first split

variable, and it is used for the construction of the temporal ordering of the partially symmetric and symmetric IDITs. After the preliminary temporal ordering has been deduced the first split variable is found, if such a variable exists. If it does not exist the IDIT is solved using a local solution method, as described in [4]. When a split variable exists the intervals are found using Algorithm 5.2. These are the intervals, which are proposed by the split variable. The IDIT is also split into partially symmetric IDITs using Algorithm 5.3. Then each of these are solved locally.
Solving a partially symmetric IDIT, as in [2] follows the same idea of solving the original IDIT, the main difference is that the preliminary temporal ordering is already found, which makes this step obsolete. The splitting of the partially symmetric IDIT continues in [3] until all split variables have been split on, thereby yielding a set of symmetric IDITs.
Until [4] the solution method has primarily focused on the global level. However, when a symmetric IDIT is deduced the solution method shifts to focus on the local solution method. As we have chosen to structure the elimination using strong junction trees, the symmetric IDIT is converted to this structure, by using Algorithms 5.4 to 5.6. When the symmetric IDIT is structured in a strong junction tree, Algorithm 5.7 is used to find an optimal policy for each decision in the symmetric IDIT, which is after the split variable resulting in the symmetric IDIT, in the temporal ordering. When the split variable is to be eliminated the strong junction tree and the optimal policies are propagated back to the partially symmetric IDIT, which is a parent in the split tree.
When a partially symmetric IDIT has received all the strong junction trees and optimal policies from its children, this information is merged using the method described in Section 5.5. Algorithm 5.7 is executed on the merged strong junction tree and the result is sent to its parent. This is continued until the root receives the results of all its children, and the merged strong junction tree, obtained from this, is solved. When all variables are eliminated the solution method returns an optimal strategy for the IDIT.

The complexity of this algorithm is exponential in the number of split variables, which in terms mean that this is not feasible for IDITs in which there are a lot of restriction functions and guards. Furthermore, as mentioned in Section 5.3, finding a minimum triangulation is also NP-hard.

To argue that the solution method in fact does solve IDITs we look at the elements of the solution method and argue for their correctness. The solution method takes outset in an IDIT, and splits this IDIT into partially symmetric IDITs. As SplitTree finds all possible splits by extracting the point in time each guard and restriction function refers to, and does this in a recursive manner until all split variables have been split on, all asymmetries are revealed and, consequently, resolved. Thus, the asymmetries can be removed in the partially symmetric and symmetric IDITs without loss of information. Therefore, information is not lost after splitting the IDIT. We can discretize the continuous variables of each symmetric IDIT to any granularity, and for each of these intervals approximate the probability using, for instance,

sampling, which can be approximated to be arbitrarily close to the exact probability distribution. We argue that the local solution of each symmetric IDIT can be done using any approach for solving influence diagrams, as an IDIT with only discrete variables is, essentially, an influence diagram. Therefore, an optimal strategy is correctly found. The merging of two symmetric IDITs has already been proven to be correct in Section 5.5. Because of this the solution method does indeed solve IDITs.

5.7 The Sampling Approach

In this section we present the sampling approach, which we use to approximate potentials including time variables. We do this by first presenting the motivation behind this approach. Then we present the general idea behind the sampling technique, we have chosen, together with an algorithm.

5.7.1 Motivation for Introducing Sampling

The motivation for choosing sampling, for determining the probability distribution for a continuous chance variable, is that we have chosen to represent time variables by χ^2-distributions. Under normal circumstances we would use an approach as described in Section 5.4. That is, when eliminating some continuous chance variable, V, the utility potential would be $\Psi^*_V = \sum_i \int f(t)dt \cdot \psi_i(V)$, where $f(t)$ is the distribution over time, t, and $\psi_i(V)$ is the utility potentials with V in their domain. In our case we cannot do this, as a χ^2-distribution is not on closed form, which means that exact integration is not a possibility. In [Broe et al., 2003] the solution to this was by numerical integration using Maple [Maplesoft, 2002]. In this thesis we have chosen to use sampling to solve the problem. We use a standard method, as described in [Gentle, 1998], for drawing random samples from a χ^2-distribution.

5.7.2 Utilizing Sampling

We approximate the probability distribution for some time variable and calculate the expected utility based on this. We calculate this by drawing a number of samples and summing the value for each sample. This sum we divide by the number of samples we have drawn, and the result is the utility for some potential. That is, for some time variable, T, we calculate $\Psi^*_T \approx \sum_j \frac{1}{n} \sum_{i=1}^{n} f_j(\mathbf{X}_i)$, where $\mathbf{X}_i$ is some random sample, n is the number of drawn samples, and f_j is the utility potentials with T in their domain. The number of samples is determined by the decision taker before sampling is performed.
When sampling we take a candidate point from some *proposal distribution* and compare that point with the target distribution, using some scheme. The nature of this scheme is elaborated on shortly.

When choosing candidate points for sampling, we use two different algorithms, depending on whether the degree of freedom is less than 0.5, or not. The proposal distribution can be any distribution, and no matter what the target distribution is, it can be proven that, given enough samples, and due to the laws of large numbers, we can approximate the target probability distribution with an arbitrary precision. The two algorithms are taken from [Gentle, 1998]. The algorithm for a degree of freedom less than 0.5 is presented in Algorithm 5.9, and for a degree of freedom greater than or equal to 0.5 we present Algorithm 5.10. For both algorithms, $\mathbf{Y}$ is the sample point, d is the degrees of freedom for the χ^2-distribution, and all other characters are just parameters to help ease the algorithms along. The two algorithms were originally constructed for use in sampling from Γ-distributions, but as a χ^2-distribution is, essentially, a special case of a Γ-distribution, we use the same algorithms. As can be deduced from the two algorithms, the proposal distribution is a uniform distribution. The scheme, we mentioned, takes on a different character through the algorithms. The lines between two **return** statements in the algorithms constitute the different schemes for manipulating numbers in order to create random samples from the χ^2-distributions.

The names of the algorithms portray the authors upon who these representations are based.

BEST/AHRENS/DIETER

1: $x = 0.07 + 0.75 \cdot \sqrt{1 - d}$
2: $b = d + \frac{exp^{-x} \cdot d}{x}$
3: **while** $i \leq n$ **do**
4: Generate u_1 and u_2 independently from $U(0, 1)$
5: $v = b \cdot u_1$
6: **if** $v \leq 1$ **then**
7: $\mathbf{Y} = x \cdot v^{\frac{1}{d}}$
8: **if** $u_2 \leq \frac{2-\mathbf{Y}}{2+\mathbf{Y}}$ **then**
9: **return** $\mathbf{Y}$
10: **else if** $u_2 \leq exp^{-\mathbf{Y}}$ **then**
11: **return** $\mathbf{Y}$
12: **else**
13: $\mathbf{Y} = \log\left(\frac{x \cdot (b-v)}{d}\right)$
14: $y = \frac{\mathbf{Y}}{x}$
15: **if** $u_2 \cdot (d + y \cdot (1 - d)) \leq 1$ **then**
16: **return** $\mathbf{Y}$
17: **else if** $u_2 \leq y^{d-1}$ **then**
18: **return** $\mathbf{Y}$
19: $i = i + 1$

Algorithm 5.9: *The algorithm for choosing a sample from a χ^2-distribution with less than 0.5 degrees of freedom. n is chosen by the decision taker.*

CHENG/FEAST

1: **while** $i \leq n$ **do**
2: Generate u_1 and u_2 independently from $U(0,1)$
3: $v = \frac{(d-\frac{1}{6\cdot d})\cdot u_1}{(d-1)\cdot u_2}$
4: **if** $\frac{2\cdot(u_2-1)}{d-1} + v + \frac{1}{v} \leq 2$ **then**
5: $\mathbf{Y} = (d-1)\cdot v$
6: **return Y**
7: **else if** $\frac{2\cdot \log u_2}{d-1} - \log v + v \leq 1$ **then**
8: $\mathbf{Y} = (d-1)\cdot v$
9: **return Y**
10: $i = i + 1$

Algorithm 5.10: *The algorithm for choosing a sample from a* χ^2*-distribution with* 0.5 *degrees of freedom, or more.* n *is chosen by the decision taker.*

The algorithms use a rejection/approval method to identify which samples should be accepted as samples. The reason for having such a method, and not just accepting every sample, is that the candidate samples are not necessarily from the correct distribution, but from some random distribution. The rejection/acceptance factor is also the reason why the different schemes, as mentioned above, are applied.

We draw our samples using either of the two algorithms presented above, according to the nature of the distribution being sampled.

Our aim is to find the utility potential obtained by marginalizing some time variable, T. To exemplify this, consider a standard χ^2-distribution with five degrees of freedom. Imagining that this represents the probability distribution for some time variable. We want to find the utility potential obtained by marginalizing this, so we need to sample from it, but beforehand we have identified intervals in which the state of this time variable changes the state space for some future decision. If we say that we have divided the time variable in two intervals, and these are $[0:5[$ and $[5:\infty[$. The utility potential obtained by marginalizing T is then:

$$\psi^*_{(}T) = \frac{1}{n}\sum_{i=1}^{n}\left(f_{0,5}(\mathbf{X}_i) + f_{5,\infty}(\mathbf{X}_i)\right), \tag{5.5}$$

where $f_{a,b}(\mathbf{X}_i)$ means the value calculated over samples, $\mathbf{X}_i$, which lie in the interval $[a:b[$, and n is the number of drawn samples. Using this approach we do not approximate the actual probability distribution, per se, but we calculate the expected value of the utility. As we only need the optimal choices for our solution, we need not know the exact probability distribution.
When marginalizing T we know that the probability potential, ϕ, is $P(T|\mathbf{X})$, where $\mathbf{X}$ is the conditioning set for T, and if this is not the case, then some other variable outside of $\mathbf{X}$ must be dependent on T. We can show that this is not the case as

all variables dependent on T, in the IDIT, have been marginalized and any other variable dependent on T would introduce an unresolved cycle in the IDIT.
If T is associated with a discrete decision variable, we find the utility potential for T as just described. If, on the other hand, T is associated with a wait decision variable we do not find the exact utility potential, but pass along the function, as shown in Equation (5.5) and when we know the optimal policy of the wait decision, we can find the utility potential for T. As long as T is not the first time variable in the IDIT it is also influenced by some other time variable. This we handle in the same manner as if T has a wait decision as a parent. That is, we pass along the function, with only the point in time represented by the time variable, which is a parent of T, as an argument. As time variables are dependent on each other, we end up sending the function along until we eliminate the last time variable. This has the unfortunate effect that the number of expressions, in the function we are sending along, rises exponentially in the number of time variables. We can put it into perspective if we consider a simple IDIT, in which there are two time variables, T_1 and T_2, such as presented in Figure 5.4.

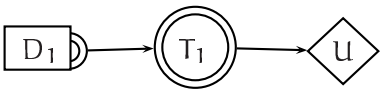


Figure 5.4: *A simple IDIT showing how the function for the utility potential grows.*

When eliminating variables from this IDIT we start with T_1, then $T^e_{D_1}$, and then D_1. Eliminating T_1 gives:

$$\psi^*_T = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i, t^e_D).$$

What we see here is that the utility potential is calculated from the distribution of T_1, but as the point in time represented by T_1 is dependent on the point in time represented by T^e_D, t^e_D, we cannot calculate the exact value. The next variable to eliminate is T^e_D, and we get:

$$\psi^*_{T^e_D} = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i, f(\mathbf{X}_j, d)).$$

That is, for every sample of the distribution for T^e_D we draw n samples over T_1. And this continues, since we see that t^e_D in fact depends on the choice of D.

In Chapter 7 we discuss the consequences we have drawn in using the approach as described in this chapter.

5.8 Summary

In this chapter we have proposed a general solution method for solving IDITs. We have shown how the split tree, which we presented in Chapter 4, is used not only as a guide for reading IDITs, but also as a guideline when solving IDITs. We have presented a method for removing asymmetries in an IDIT. Besides this we have shown how strong junction trees can be used to solve IDITs, while leaving the door open for other approaches.
Because of our definition of time in IDITs we have come across a problem, namely that of using χ^2-distributions to represent the density function for time variables. We have proposed using a sampling technique, which is based on algorithms by Best, Ahrens, and Dieter and the other by Cheng and Feast.
We have, however, reached the conclusion that using sampling only introduces a new problem, as our solution method ends up having to deal with exponentially large functions in the number of time variables in the IDIT we are solving.

Chapter 6

Results and Discussion

In this chapter we begin by illustrating the use of the solution method. We apply it on two IDITs designed to point out some of the different elements of IDITs, and how these elements affect the process of solving an IDIT. In Section 6.2, we compare elements of the solution method to using other approaches, such as a nonuniform discretization of continuous variables [Kozlov and Koller, 1997], and using a multi-stage Monte Carlo approach [Charnes and Shenoy, 2002]. Finally, in Section 6.3 we discuss IDITs as a framework.

6.1 Solving Two Examples

In this section we present two examples. These examples incorporate some of the aspects of time we have discussed in Chapter 4, and we use them to exemplify the solution method, we devised in Chapter 5. The goal of applying the solution method to an IDIT is to find the optimal strategy for the decisions in that IDIT.

6.1.1 Example One - Post-Realized Utility Function and Chance Variables Dependent on Time

In the first example we present an instance of a post-realized utility function, a discrete chance variable dependent on time, and a time variable associated with a decision. The main focus of this example is the local solution method, that is, the elimination of variables, both continuous and discrete. The IDIT, we use as a model for this, is depicted in Figure 6.1.

For this IDIT the chance variable A has a marginal probability distribution, B has a conditional probability distribution, $P(B|A)$, and C has a conditional probability distribution, $P(C|B, T_D^e)$, in which the condition set consists of the discrete chance variable, B, and the time variable, T_D^e. The two time variables, T_D^e and T have,

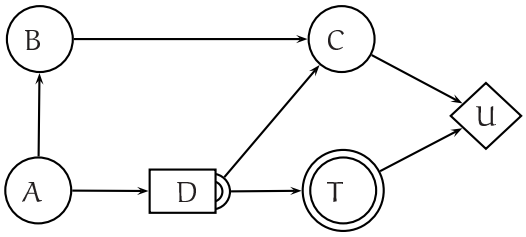


Figure 6.1: *A model of an IDIT containing a post-realized utility function and a time dependent chance variable.*

per definition, a conditional probability distribution defined by a χ^2-distribution, with the degrees of freedom determined through their discrete parents. As T has no discrete parents, we set it to three degrees of freedom, which we have chosen as the default.
The three chance variables are binary variables, and the tables representing their probability distributions are presented in Tables 6.1(a) and 6.1(b).

a_1	a_2
0.3	0.7

(a)

		A	
		a_1	a_2
B	b_1	0.2	0.6
	b_2	0.8	0.4

(b)

Table 6.1: *(a): The marginal probability distribution,* $P(A)$*, for* A*. (b): The conditional probability distribution,* $P(B|A)$*, for* B*.*

Table 6.2 shows the parameters for C, given the configuration of its discrete parent. The function for representing C is $f(p,t) = (1-p)^t$ for $C = c_1$ and $f(p,t) = 1-(1-p)^t$ for $C = c_2$, where p is the parameter we find in Table 6.2, and t represents time.

		B	
		b_1	b_2
C	c_1	0.03	0.05
	c_2	0.03	0.05

Table 6.2: *The table of parameters for the time dependent chance variable,* C*.*

The only decision variable for this IDIT is D. It has the states d_1 and d_2. Taking choice d_1 results in a timed action, which takes 10 time units to perform, yielding a distribution for T_D^e, which is displaced by 10 time units and the distribution has 2.8 degrees of freedom. If the choice chosen is d_2, then the distribution for T_D^e is

displaced by 15 time units and the distribution has 3.4 degrees of freedom. The two density functions are depicted in Figure 6.2.

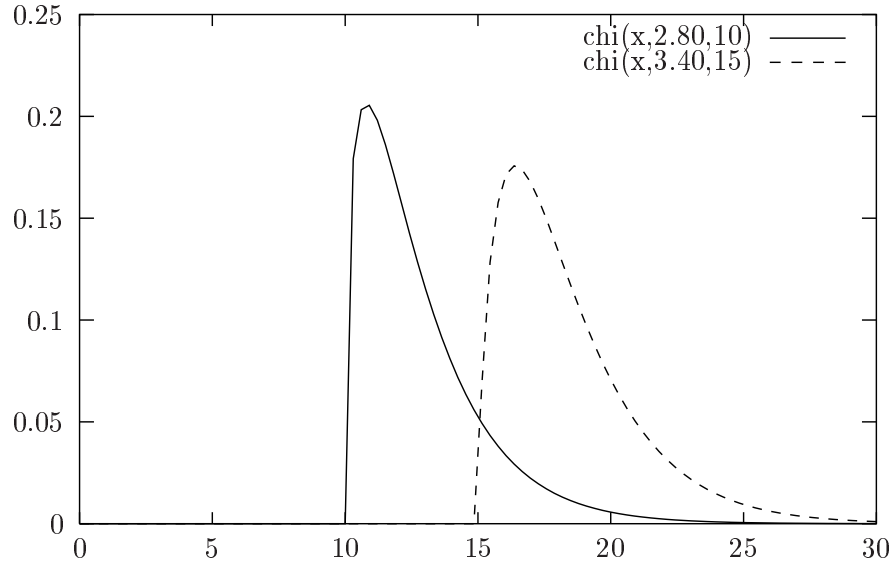


Figure 6.2: *The density function for* $\mathsf{T}^{e}_{\mathsf{D}}$ *given the choice of* D.

The local utility function for this IDIT, U, is defined as:

$$\mathsf{U}(\mathsf{C} = \mathsf{c}_1, \mathsf{T}) = \frac{\log \mathsf{t}}{4}$$

$$\mathsf{U}(\mathsf{C} = \mathsf{c}_2, \mathsf{T}) = \frac{2 \cdot \log \mathsf{t}^2}{\mathsf{t}}.$$

In order to solve the IDIT presented in Figure 6.1, we first determine that no more splits are needed. Therefore, following the solution method we construct a strong junction tree for the IDIT. The strong junction tree respects the reverse of the temporal order. To find the temporal order, we first find the preliminary temporal order of the IDIT, this order is $\mathsf{A} \prec^{\rightarrow}_{\mathsf{t}} \mathsf{D} \prec^{\rightarrow}_{\mathsf{t}} \mathsf{T}^{e}_{\mathsf{D}} \prec^{\rightarrow}_{\mathsf{t}} \mathsf{T} \prec^{\rightarrow}_{\mathsf{t}} \{\mathsf{B}, \mathsf{C}\}$. There are no splits in this IDIT, yielding the preliminary temporal order as the temporal order of the IDIT. The moral graph is then constructed and the strong triangulation is performed. We use heuristics to find that the elimination order should be C before B. If we were to eliminate B first, we must put in two fill-ins, while eliminating C results in only inserting one fill-in. In constructing the moral graph two edges are entered, these are $(\mathsf{B}, \mathsf{T}^{e}_{\mathsf{D}})$ and (C, T). In performing the strong triangulation four fill-ins are entered. These are: (B, T), $(\mathsf{A}, \mathsf{T}^{e}_{\mathsf{D}})$, (A, T), and (A, D). The resulting graph is depicted in

Figure 6.3(a). From the triangulated graph the cliques are identified, and a strong junction tree is constructed. Figure 6.3(b) illustrates the strong junction tree for the graph. We illustrate the separator sets as boxes.

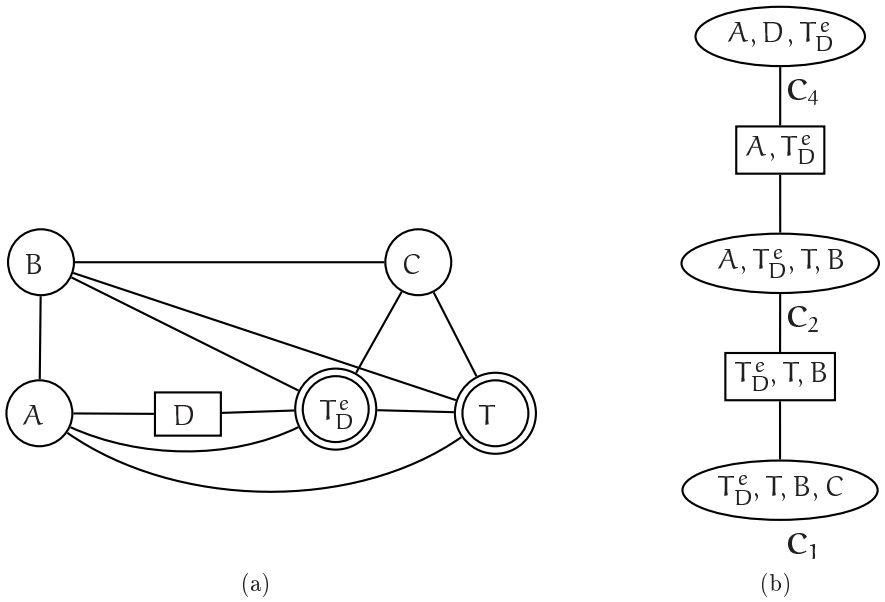


Figure 6.3: *(a): The triangulated graph for the IDIT of Figure 6.1. (b): A strong junction tree for the graph to the left.*

The cliques of the strong junction tree we name $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_4$, where $\mathbf{C}_4$ is the strong root. These are the names of the bottom clique, the middle clique, and the top clique, respectively. The sets of probability and utility potentials for the entire strong junction tree are: $\mathbf{\Phi} = \{P(A), P(T^e_D|D), P(T|T^e_D), P(B|A), P(C|B, T^e_D)\}$ and $\mathbf{\Psi} = \{U(C, T)\}$, respectively.

As mentioned above, the first variable we are eliminating is C. This is in accordance to the rules for message passing and absorption as described in Chapter 5.
The set of probability potentials attached to $\mathbf{C}_1$, $\mathbf{\Phi}_{\mathbf{C}_1}$, is: $\{P(C|B, T^e_D)\}$. And the set of utility potentials, $\mathbf{\Psi}_{\mathbf{C}_1}$, is: $\{U(C, T)\}$. The utility function for this utility potential is described above. Updating the probability potentials is straight forward as $P(C|B, T^e_D)$ is a unit potential, meaning no other variables are dependent on C, and therefore the potential is simply removed, yielding $\mathbf{\Phi}^* = \{P(A), P(T^e_D|D), P(T|T^e_D), P(B|A)\}$. $\mathbf{\Psi}^*$ is then found to be $\{U(B, T^e_D, T)\}$. The utility potential, ψ^*_C, is defined as $\sum_C U(C, T) \cdot P(C|B, T^e_D)$. This gives us the following utility functions, given the configuration of the domain of U:

$$U(B = b_1, T, T^e_D) = \left(\frac{\log t}{4} \cdot (1 - 0.03)^{t^e_D}\right) + \left(\frac{2 \cdot \log t^2}{t} \cdot (1 - (1 - 0.03)^{t^e_D}\right) \text{ and}$$

$$U(B = b_2, T, T_D^e) = \left(\frac{\log t}{4} \cdot (1 - 0.05)^{t_D^e})\right) + \left(\frac{2 \cdot \log t^2}{t} \cdot (1 - (1 - 0.05)^{t_D^e})\right),$$

where t is the point in time represented by T, and t_D^e is the point in time represented by T_D^e.

After passing the message of the potentials along, this gives us a strong junction tree as depicted in Figure 6.4.

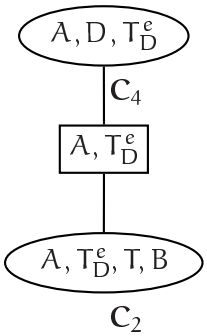


Figure 6.4: *The strong junction tree after eliminating* C *and passing along the messages.*

The sets of potentials of the absorption of $\mathbf{C}_1$ in $\mathbf{C}_2$ are, $\mathbf{\Phi}_{\mathbf{C}_2} = \{P(B|A), P(T|T_D^e)\}$ and $\mathbf{\Psi}_{\mathbf{C}_2} = \{U(B, T_D^e, T)\}$. Following the elimination order, the next variable to be eliminated is B. Again the probability potential is a unit potential and can simply be removed, yielding $\mathbf{\Phi}^* = \{P(A), P(T_D^e|D), P(T|T_D^e)\}$. The set of utility potentials is then updated to be $\mathbf{\Psi}^* = \{U(A, T_D^e, T)\}$ after elimination of B. The utility function for ψ_B^* is found as $\sum_B U(B, T_D^e, T) \cdot P(B|A)$. This yields the following utility functions, given the configuration of the domain:

$$U(A = a_1, T, T_D^e) = (U(B = b_1, T, T_D^e) \cdot 0.2) + (U(B = b_2, T, T_D^e) \cdot 0.8)$$

and

$$U(A = a_2, T, T_D^e) = (U(B = b_1, T, T_D^e) \cdot 0.6) + (U(B = b_2, T, T_D^e) \cdot 0.4).$$

Now T is to be eliminated. This gives us that $\mathbf{\Phi}^* = \{P(A), P(T_D^e|D)\}$, $\mathbf{\Phi}_{\mathbf{C}_2} = \{P(T|T_D^e)\}$, and $\mathbf{\Psi}^* = \mathbf{\Psi}_{\mathbf{C}_2} = \{U(A, T_D^e)\}$. The utility function for ψ_T^* is found as $\sum_T U(A, T_D^e, T) \cdot P(T|T_D^e)$. This yields the following utility function, given the configuration of the domain:

$$U(A = a_1, T_D^e) = \int_T U(A = a_1, t, T_D^e) \cdot \chi^2(t - t_D^e)(3.00)dt \text{ and}$$

$$U(A = a_2, T_D^e) = \int_T U(A = a_2, t, T_D^e) \cdot \chi^2(t - t_D^e)(3.00)dt,$$

where $\chi^2(t - t^e_D)(3.00)$ is the χ^2-distribution, which has three degrees of freedom, and is displaced by a measure of t^e_D, where t^e_D is the point in time represented by T^e_D.
As we have said earlier, we use sampling to handle the continuous chance variables. This gives that the utility potential after eliminating T is:

$$U(A = a_1, T^e_D) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i) \text{ and}$$

$$U(A = a_2, T^e_D) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i),$$

where $f(\mathbf{X}_i)$ is a function returning the value of the utility function given some sample, $\mathbf{X}_i$, and n denotes the number of samples drawn.

The function we have gotten, we pass up the tree, and the resulting strong junction tree is depicted in Figure 6.5.

Figure 6.5: *The strong junction tree after eliminating* T *and absorbing* $\mathbf{C}_2$.

The next variable to be eliminated is T^e_D, which is the first time variable of the IDIT and, consequently, the last time variable to be eliminated. Again only one probability potential has the variable to be eliminated in its domain, so the potential is simply removed and $\mathbf{\Phi}$ updated accordingly, yielding $\mathbf{\Phi}^* = \{P(A)\}$. $\mathbf{\Psi}^*$ is updated to $\{U(A, D)\}$. $\psi^*_{T^e_D}$ is found by $\sum_{T^e_D} U(A, T^e_D) \cdot P(T^e_D|D)$. This yields the following utility functions, given the domain:

$$U(A = a_1, D = d_1) = \int_{T^e_D} U(A = a_1, T^e_D) \cdot \chi^2(t^e_D - 10)(2.80)dt^e_D,$$

$$U(A = a_1, D = d_2) = \int_{T^e_D} U(A = a_1, T^e_D) \cdot \chi^2(t^e_D - 15)(3.40)dt^e_D,$$

$$U(A = a_2, D = d_1) = \int_{T^e_D} U(A = a_2, T^e_D) \cdot \chi^2(t^e_D - 10)(2.80)dt^e_D \text{ and}$$

$$U(A = a_2, D = d_2) = \int_{T^e_D} U(A = a_2, T^e_D) \cdot \chi^2(t^e_D - 15)(3.40)dt^e_D.$$

The displacement of the χ^2-distributions, which for these is set to 10 and 15, originates from the definition of D.

Now we draw samples for T^e_D. This gives us a utility for $\psi^*_{T^e_D}$ as:

$$U(A = a_1, D = d_1) = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i, f(\mathbf{X}_j, 10)),$$

$$U(A = a_1, D = d_2) = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i, f(\mathbf{X}_j, 15)),$$

$$U(A = a_2, D = d_1) = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i, f(\mathbf{X}_j, 10)), \text{ and}$$

$$U(A = a_2, D = d_2) = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i, f(\mathbf{X}_j, 15)),$$

where m is the number of drawn samples for T^e_D, and $f(\mathbf{X}_j, d)$ is a function returning the value of the utility function given sample $\mathbf{X}_j$ and the point in time associated to choice, d, of D.

The next variable to be eliminated is D, which is a decision variable. The method of eliminating a decision variable is different than for chance variables. The utility potential is updated by choosing the maximum over a set of expected utilities, instead of summing over it. This is because we aim at finding the optimal strategy, which is the strategy yielding the maximum expected utility.
When eliminating D from $\mathbf{C}_4$, the sets of potentials are updated. As D affects no remaining chance variables $\mathbf{\Phi}^*$ is unaffected. $\mathbf{\Psi}^* = \{U(A)\}$. $U(A)$ is found by $\underset{D}{\operatorname{argmax}}\, U(A, D)$. This yields the following utility function, given the domain:

$$U(A) = \underset{D}{\operatorname{argmax}}\, U(A, D).$$

The object is then to determine the optimal strategy, $\hat{\Delta}$, for the IDIT. As only one decision exists in the IDIT, this is found by considering the policy for this decision alone.
As can be seen from the example, the size of the function grows exponentially in the number of time variables. If, for instance, the number of samples for the first time variable is set to 1000, and the same is done for the second time variable, the number of expressions in the function is in the range of 10^6. As a sample size of 1000 is rather small, we can conclude that solving IDITs using this type of sampling approach is intractable. Even though this means that this approach is practically unusable for problems with more than a few time variables, we are still able to use it to clarify how an ideal solution of an IDIT is constructed.
During the example we have used sampling to resolve the occurrences of integrals. Sampling can be performed by use of tools such as WinBUGS [MRC Biostatic Unit, Cambridge, UK,], we have, however, not had time to

explore this path. Thus, we cannot determine the actual policy for taking D. What is needed for finding this policy, however, is a numerical solution, as the functions for determining it are already given.

6.1.2 Example Two - Split and Wait Decision

In this second example we focus on the creation of a split tree and the merging of leaf nodes in the split tree. As can be seen, Figure 6.6 contains guards, and in particular, guards in relation to a cycle. Furthermore, the IDIT contains a wait decision.

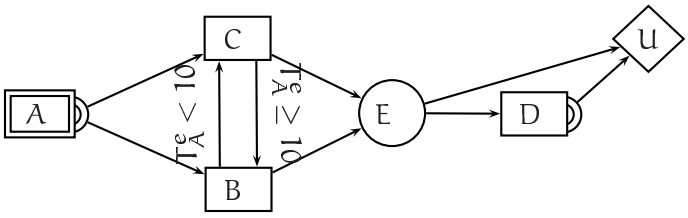


Figure 6.6: *A model of an IDIT containing guards, introducing the need to split on the first time variable.*

For the IDIT represented in Figure 6.6, the state spaces of the variables are as follows. A is a wait decision, meaning that it has a continuous state space. The decisions B, C, and D are all binary decisions, resulting in the choices b_1, b_2; c_1, c_2; and d_1, d_2, for B, C, and D, respectively. The choices in D result in timed actions lasting five and ten time units, respectively.
E, is a discrete chance variable in the IDIT, has the conditional probability distribution, $P(E|C,B)$, described in Table 6.3.

		B			
		b_1		b_2	
		C		C	
		c_1	c_2	c_1	c_2
E	e_1	0.3	0.4	0.6	0.7
	e_2	0.7	0.6	0.4	0.3

Table 6.3: *The conditional probability distribution for* E*, given* B *and* C*.*

There are two time variables in the IDIT, T^e_A and T^e_D. T^e_A has no discrete parents, yielding a probability distribution with three degrees of freedom, and a displacement dependent on the choice of A. T^e_D has one discrete parent. The degrees of freedom in the χ^2-distribution for T^e_D is therefore dependent on the choice of D. If $D = d_1$, T^e_D has 4.6 degrees of freedom, and a displacement of $t^e_A + 5$. And if $D = d_2$, T^e_D has 4.2 degrees of freedom and a displacement of $t^e_A + 10$, where t^e_A is the point in time

represented by T^e_A.
There is one local utility function, U, in the IDIT. The function of U is dependent on both the outcome of E and the time represented by T^e_D. The function is defined as:

$$U(E = e_1, T^e_D) = \frac{\log t}{4}$$

$$U(E = e_2, T^e_D) = \frac{2 \cdot \log t^2}{t}.$$

Examining the IDIT of Figure 6.6, we see that guards are on two of the arcs. When solving the IDIT we construct a split tree. The split tree is built from the preliminary temporal ordering of the IDIT. For this IDIT this is deduced to be:

$$A \prec^{\rightarrow}_t T^e_A \prec^{\rightarrow}_t \{B, C\} \prec^{\rightarrow}_t E \prec^{\rightarrow}_t D \prec^{\rightarrow}_t T^e_D$$

, showing that the split is from the first time variable, as this is the last time variable before a set of unordered decisions occurs. The split tree is found by splitting on T^e_A, is presented in Figure 6.7.

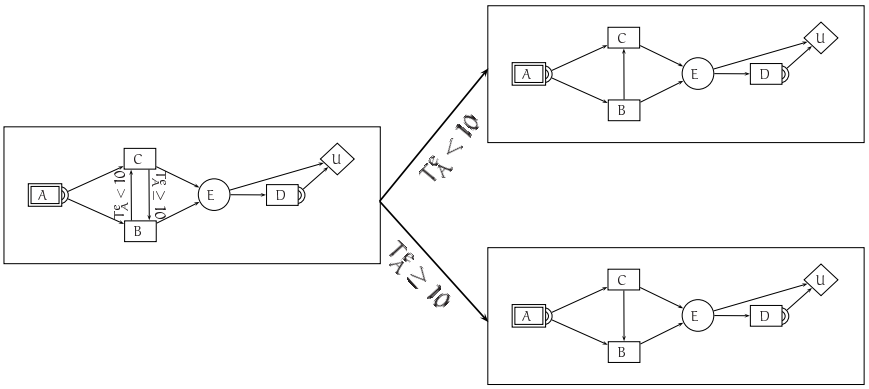
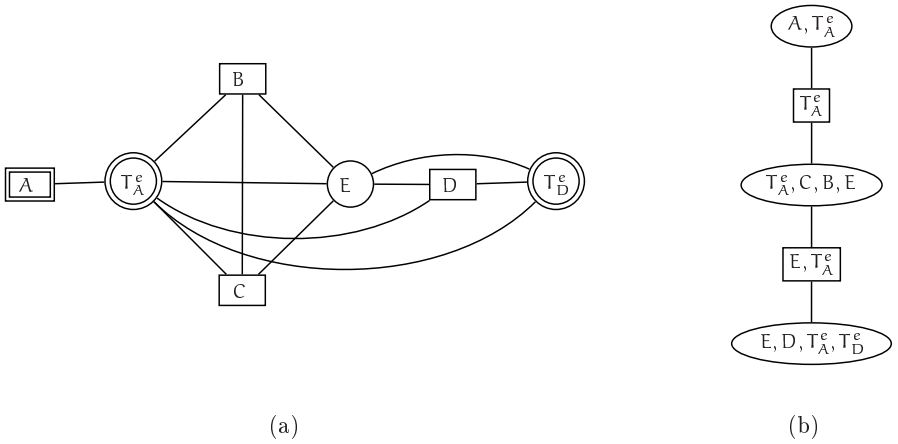


Figure 6.7: *The split tree for the IDIT presented in Figure 6.6.*

We construct the split tree by placing the original IDIT in the root of the tree. Then we find the first split variable and identify the restriction functions referring to it. For this IDIT those are the guards on the arcs connecting B and C. We construct a node for each partially symmetric IDIT resulting from the split, and remove arcs, for which the guard evaluates to false. Then we find the preliminary temporal ordering for each partially symmetric IDIT and identify new potential split variables. No such variable exist, and the partially symmetric IDITs are identified to be symmetric IDITs.

A strong junction tree for each symmetric IDIT, is now created, beginning with the one depicted in the top leaf node. This symmetric IDIT has, because of the split, the temporal ordering: $A \prec^{\rightarrow}_t T^e_A \prec^{\rightarrow}_t B \prec^{\rightarrow}_t C \prec^{\rightarrow}_t E \prec^{\rightarrow}_t D \prec^{\rightarrow}_t T^e_D$, this gives only one possible elimination order. The resulting triangulated graph and strong junction

tree, resulting from this elimination order, are depicted in Figures 6.8(a) and 6.8(b), respectively. These are constructed in a similar manner as for the previous example.



(a) (b)

Figure 6.8: *(a): The triangulated graph for the IDIT of the leaf nodes of Figure 6.7. (b): A strong junction tree for the graph presented to the left.*

During construction of the strong junction tree, edges are added in the moralization of the symmetric IDIT, and fill-ins are added in the strong triangulation of the moralized graph. The edges are: (T^e_A, B), (T^e_A, D), (B, C), and (E, T^e_D) for the moralization, and the fill-ins are: (T^e_A, E), (T^e_A, C), and (E, D) for the strong triangulation.
Examining the symmetric IDITs of the split tree further, we notice that their triangulated graphs are equivalent, thus they result in equivalent strong junction trees. Therefore, we do not show the strong triangulation or the strong junction tree for the symmetric IDIT in the lower leaf of the split tree.

The method for solving the strong junction tree follows the method, which we went through in the first example. We, therefore, only show what is done in the presence of a split variable and when eliminating a wait decision.

We do not show the elimination of the variables before T^e_A in the elimination ordering, as this resembles the elimination done in the first example.

The elimination of variables up until T^e_A in the two symmetric IDITs results in equivalent sets, $\mathbf{\Phi}$ and $\mathbf{\Psi}$ but with different values for the potentials in $\mathbf{\Psi}$, for the two symmetric IDITs. Before eliminating T^e_A the two strong junction trees are similar in structure, so the merged strong junction tree is a clique consisting of A and T^e_A. The set of probability potentials for this clique consists of only $P(T^e_A|A)$. This is the case for both symmetric IDITs. The set of utility potentials consists of a function, U, over the domain T^e_A. U is a function, which, as long as T^e_A is less than ten, equals

U_1. U_1 is the utility potential for the upper symmetric IDIT. U equals U_2 for values of T_A^e greater than or equal to ten, where U_2 is the utility potential for the lower symmetric IDIT. There is in fact no reason to distinguish between U_1 and U_2, when solving the IDIT, as $\underset{D}{\mathrm{argmax}}(\underset{D'}{\mathrm{argmax}}\ \psi)$ is equivalent to $\underset{D'}{\mathrm{argmax}}(\underset{D}{\mathrm{argmax}}\ \psi)$ for any two decisions, D and D'.

Eliminating T_A^e we get $\mathbf{\Phi}^*_{T_A^e} = \varnothing$ and $\mathbf{\Psi}^*_{T_A^e} = \sum_{T_A^e} U(A) \cdot P(T_A^e|A)$. This yields the following function over A:

$$U(A) = \int_0^{10} U_1(T_A^e) \cdot \chi^2(t_A^e - a)(3.00)dt_A^e + \int_{10}^{\infty} U_2(T_A^e) \cdot \chi^2(t_A^e - a)(3.00)dt_A^e,$$

where a denotes the time chosen for A.

To eliminate A we maximize over $U(A)$. As A is a wait decision, we find the maximum by computing the derived of $U(A)$ and finding the extrema. Then we compare these extrema to find the global maximum. If two or more maximum points exist, we choose the first, with respect to time.

This concludes the two examples of how we solve IDITs.

6.2 Alternative Approaches

In the previous section we saw how two examples were solved using the solution method we present in Chapter 5.

In this section we discuss alternative approaches handling different elements of the solution of IDITs. As we saw in the first example, the use of sampling as proposed introduces a grand complexity of the functions being sampled. In this section we discuss alternatives to that approach. We look at [Kozlov and Koller, 1997], in which a nonuniform discretization for the discrete variables is used. Then we discuss an approach presented in [Charnes and Shenoy, 2002], which utilizes a sampling method, in which samples are drawn from multiple variables, and the aim is to approximate the optimal policy for decision variables. We discuss numerical integration, as this is the approach used in [Broe et al., 2003]. Finally, we mention an approach in which the distributions of the time variables is approximated using polynomials.

6.2.1 Discretizing Continuous Variables

Another way, than using sampling to approximate the probability distribution of time variables, would be to discretize these variables. Usually one would go about this by discretizing variables independently of each other and one at a time. In [Kozlov and Koller, 1997] a method, for discretizing continuous variables in a hybrid Bayesian network in a nonuniform and dynamic manner, is proposed. They

present what they call a Binary Split Partition tree, a BSP tree for short, which they use for discretizing variables in cliques of a junction tree. A BSP tree is simply a data structure for storing information regarding how the discretization of some interval is performed.
The discretization is performed in iterations. Each iteration bases the new intervals on the nature of the distribution in each interval. These iterations continue until either the decision taker chooses not to have any more iterations, or the Kullback-Leibler distance, which is the relative entropy, between the original distribution and the discretized distribution is smaller than some set value. This value is set by the decision taker.

For each iteration, the algorithm is designed to concentrate intervals around areas of the distribution, which has the most activity. That is, considering two ranges, $[a : b[$ and $[b : c[$, where a and c are arbitrary reals, and b is the mean of a and c, the algorithm would discretize the interval, which has the most fluctuation in values, more thoroughly than the interval with the least fluctuation of. The value for a discretized interval is the mean for the same interval in the original distribution. The reason for this is that this ends up giving the minimal Kullback-Leibler distance, KL-distance, between the two distributions. This is not proven in [Kozlov and Koller, 1997], but they refer to [Cover and Thomas, 1991] for the proof and justification for using the Kullback-Leibler distance as a guidance measure.
The reason for concentrating intervals around active parts of the distribution is that this should help in making the errors introduced by any form for discretization smaller and therefore give a closer approximation with fewer operations.

This only describes the nonuniform part of the method. The dynamic part concerns how the variable, which should be discretized, is chosen. Before any variable is discretized the hybrid Bayesian network is converted to a junction tree. The discretization is stored in the BSP tree, which is organized, so the original function is the root. Each node may have two children, each child representing half the distribution of its parent. When choosing which half to discretize, gradient descent is used, in order to discover the activity of the function in the interval currently being examined. Finding which variable to discretize is done by considering the KL-distance between the joint probability distribution of the continuous variables in the current clique and the joint probability distribution of the same variables after a discretization. The discretization yielding the minimum KL-distance between these two distributions, is then chosen. When performing many discretizations, this quickly becomes an intractable approach and heuristics are applied to find the variable and interval to discretize.
We do not go through the algorithms or proofs in this thesis, but refer the reader to [Kozlov and Koller, 1997] for further detail.

If we were to use this approach on an IDIT, and still using the split trees, we would have some intervals, which are predefined. This ruins part of the approach, as this leads the use of a BSP tree inapplicable. A similar data structure can be constructed,

which takes into consideration the intervals imposed by the splits in the split tree. The algorithm must then also consider these intervals when discretizing the time variable. As a result of the semantics of time variables, discretization of a time variable, results in intervals of the form $[a + t : b + t[$, where a and b are the limits of the interval if the time variable was not dependent on any other variable, and t is the latest point in time represented by the variables influencing this time variable. A further restriction on the time variables is their range. A time variable may not have an infinite range, if this method is to be utilized for discretizing it, as each iteration of the discretization splits the considered in two equally large intervals.
As long as no wait decision variables are in the IDIT the dynamic aspect of the approach proposed in [Kozlov and Koller, 1997] is still applicable. If a wait decision is to be discretized, a policy for this wait decision variable should be devised, and then the decision could be converted to a chance variable in a manner such that the probability distribution for this chance variable respects the policy of the wait decision. This would then represent a discretized chance variable.

6.2.2 Multi-stage Monte Carlo using Local Computation

Multi-stage Monte Carlo [Charnes and Shenoy, 2002], or MMC for short, differs from our approach in the way variables are eliminated and how sampling is performed. Furthermore, MMC requires that all continuous variables are discretized.
MMC was devised for use on influence diagrams and by a discretization and the imposed resolutions of asymmetries could be applied to IDITs. The purpose of MMC is to approximate optimal policies for decisions in situations, where the potentials of the influence diagram grow so large that it is intractable to calculate their exact values.
In principle, MMC works by, for each configuration of the required past of some decision, D, sampling the variables influencing the utility, influenced by D. Multiple samples are taken and when some threshold is reached, no more samples are drawn, and the configuration, yielding the optimal policy, is chosen. Before taking samples, all configurations are inspected, and invalid configurations are not considered. A threshold, for the number of samples to draw, could be some number chosen by the decision taker, or, as proposed in [Charnes and Shenoy, 2002], an $(\varepsilon, \alpha)-$approximation. The limits for this approximation are set as a function of the number of samples drawn. This is a way of approximating the optimal policy within ε of the maximum expected utility, with a level of confidence of $100(1-\alpha)\%$.
A strategy for an influence diagram is (ε, α)-optimal, if $P(E_j) = 1-\frac{\alpha}{k}$, for $j = 1, \ldots, k$, where E_j is the event in which "the decision function selected in stage j has expected utilities within ε of the corresponding expected utilities of an optimal decision function for that stage". When an optimal policy has been approximated for some decision variable, that decision variable is converted to a decision function respecting the optimal policy. The stage refers to which decision variable is being considered.
Examining if the expected utility is within ε of the maximum expected utility is

done by letting F_{ij} be the event that "the ith estimated expected utility at stage j is within ε of its true value". This means that if the ith estimate follows this definition, then $P(E_j) \geq P(\cap_{i_1}^{n_j} F_{ij}) \geq 1 - \frac{\alpha}{k}$ for $j = 1, \ldots, k$.
We do not explain this any further but refer the reader to [Charnes and Shenoy, 2002] for further discussion.

For MMC the first thing to do is to discretize any continuous variables, this means that both time variables and wait decision variables are discretized. As this approach is developed for use on influence diagrams, the construction of a split tree, to handle asymmetries, is still a valid step. As all variables are discrete, MMC can be utilized on each of the symmetric IDITs of the split tree. We need to do this for each symmetric IDIT, as the resolved asymmetries result in different IDITs, possibly yielding different optimal policies for the decisions. This could be that some chance variable is observed before taking some decision in one symmetric IDIT, while being observed before taking another decision in another symmetric IDIT, because of guarded arcs. This approach handles elimination with outset in the last decision in some influence diagram. When enough samples are drawn, the approximated optimal policy is used to convert that decision variable to a chance variable. This variable is always in the state yielding the maximum expected utility given the configuration of its conditioning set.
When the decision variable, D, is converted to a chance variable, the next decision variable is found, and the local utility functions, influenced by D, are eliminated. A new local utility function, which portrays the numbers found to be the maximum expected utilities for D, is constructed. Now, because of the elimination of some local utility functions, there might be barren nodes. Such nodes are removed from the influence diagram, as they yield no useful information for the solution [Shachter, 1986].

Examining the definition of IDITs we find some problems though. First there is the discretization of variables, this always results in some measure of error, and the less discretizations the greater the error. Of course there is no rule, which says that the discretization may not result in an arbitrarily large number of intervals.
Another aspect, which could prove to be a problem, also has to do with discretization, it arises if the IDIT, we are solving, contains post-realized utility functions. There is, usually, no upper limit for the extent of the time variable in a post-realized utility function. This means that it, in principle, ranges from the end-time of the last time variable to infinity. This makes it hard to have an intelligent manner of discretizing it without the risk of loosing much information, no matter how fine a granularity the discretization uses. We can get around this problem by saying that from some point in time and forward, the time variable has a zero probability, or simply just to set a limit on the range of the time variables.

6.2.3 Numerical Integration

Using numerical integration is another approach to solving the question of probability distributions for time variables. [Broe et al., 2003] used this method through Maple [Maplesoft, 2002]. Using numerical integration, the distribution is discretized to a number of uniform intervals. The idea is then to approximate the value for each interval and through this approximating the probability distribution for the entire distribution. This is done by, for each of these intervals using the mean of the original distribution as the value for the interval. The finer the discretization, the closer the approximation.
The difference between this and the discretization method described in Section 6.2.1 is that this approach does not take into consideration that some intervals have a higher density than others, nor does it discretize in a dynamic manner.

6.2.4 Approximation using Polynomials

One approach to approximating a density function is to approximate its behaviour with a new function. This can be done in different manners, but one approach is to use Tailor series, as proposed in [Nielsen, 2003]. The idea is to approximate the probability distributions with polynomials, which are both integratable and differentiable. As summation and multiplication for these still are polynomials with these properties finding the approximated maximum expected utility can be done easily.

6.3 Discussion of the Framework

The goal of this project has been to develop a framework for DPITs. From the beginning three key requirements, which the representation language should fulfill, were set up. These requirements specify that the representation language should model DPITs in a manner which is compact, easy to read, and unambiguous. [Broe et al., 2003] had a fourth requirement, namely that the representation language should be complete with respect to modelling DPITs. We have removed this requirement as it is impossible to fulfill unless the class of DPITs has been precisely defined, thus yielding a new representation language.
Even though the fourth requirement has been removed we have still sought to make a framework, which has as much expressive power as possible. In this thesis we have extended IDITs to also handle situations where the quantitative part of time influences the order of taking decisions. Furthermore, we have made it possible to model uncertainties of time not related to decisions, by introducing post-realized utility functions. However, by these additions the representation language has become increasingly complex, and we stand the risk that models of DPITs therefore end up being harder to comprehend.
In this section we discuss precisely this dilemma in relation to IDITs and try to lay

bare the consequences of the choices we have taken. In doing this we hope to give future researchers in the area of decision problems involving time a foundation on which to base their choices.

The expressive power of IDITs aims at expressing all possible DPITs. Therefore, all aspects found, which relates to time, have been sought expressible in IDITs. This has been done in order to give the modeller of DPITs as much freedom as possible, and give him a universal tool for communicating and solving DPITs.
Another approach is to try to classify the aspects of time in logically connected classes, and make a specialized framework for each such class. This yields a framework, which is not capable of modelling all aspects of time, but models some in a precise and compact manner. As noted before, time introduces asymmetries in decision problems. These aspects are good candidates for classes of DPITs. That is, one class handling those DPITs, in which the quantitative aspect of time restricts the state space of decision variables, changes the set of observed variables, or imposes different orderings of decisions. Another class then consists of those DPITs, for which time influences the states of variables, and has an influence on the preferences of the decision taker. Both classes should fulfill the requirements with respect to representing time as a continuous element. They could, however, do this in different manners. The expressive power of two frameworks modelling these classes of DPITs would each be less than that of IDITs, however, as the models from these frameworks would be specialized to show the important information relating to the class directly in the graphical representation. For instance, a representation language without asymmetries may use dashed arcs to point out an influence of time, whereas in IDITs we have chosen to use that form of representation to tell the reader that there exists a restriction between the two variables connected by the dashed arc. Likewise, the solution method for any of the two classes would be faster as some steps of the solution method for IDITs would be obsolete. The splitting in order to reveal asymmetries would not be necessary for one of the frameworks, and the fact that time does not influence probabilities and utility functions in the other specialized framework, could be utilized to perform a discretization of time in some manner.

The choice of whether a specialized framework is preferred to a general purpose framework, as IDITs, or not, is subjective. If models should reflect a complex real world problem, and the primary goal of this model is to find an optimal strategy for taking the decisions, the more expressive framework would be chosen. This framework can model all aspects of the problem, and an optimal strategy is given based on all these aspects. However, when the model only needs to represent a part of a complex problem, as could often be the case, a specialized framework is preferred. This mounts to a question about the decision problems being modelled, and the outcome of modelling. If most decision problems involving time only include one of the two classes of DPITs, the specialized frameworks would be preferred. If, on the other hand, most decision problems span both of the specialized frameworks, the general framework would be preferred. To conclude on this, more research should be applied into the influence of time. Furthermore, the lack of expressive power in a

specialized framework, might be less important if the framework is easier to read. Looking at the three requirements set up for the framework, it can be seen that IDITs is a compact and unambiguous framework. Whether or not, IDITs are easy to read is a subjective discussion, but as influence diagrams are usually deemed hard to grasp, IDITs are most likely also hard for a layman to grasp.

6.4 Summary

We have applied the solution method, as proposed in Chapter 5, to two different examples. The examples incorporate the use of both a cycle and an instance of a post-realized utility function. Besides this we have presented alternatives and modifications to the sampling approach and the solution as a whole, such as the use of a discretization for the continuous variables.
We have concluded this section by discussing IDITs as a framework. We have discussed the problems we have identified in the structure as it is now and have proposed a way of splitting it to a set of specialized frameworks.

Chapter 7

Conclusion and Future Work

The aim of this project has been to represent and solve DPITs. We have done this by extending the representation language, IDITs, proposed by [Broe et al., 2003], in order to be more expressive, and by devising a solution method for IDITs.

By analysis of DPITs we have found additional requirements for a framework modelling them. These requirements ensure that the framework can handle the ordering of decisions being influenced by time and post-realized utility functions. The latter being the cases where a utility function is not realized immediately after the timed action of the last decision is executed, but at some later point in time. An example of this could be the event of selling stocks, as the payoff of selling is realized the next day, thus the market value of the stock may have fallen since the decision to sell was taken.

We have refined IDITs to be capable of handling these additional requirements by allowing guarded cycles between decisions, and by adding the possibility of having time variables, which are not associated with decisions.
Furthermore, we have shown that a temporal ordering of variables in an IDIT exists and how this temporal ordering can be deduced from the two levels of the IDIT. This is also used to show that IDITs are welldefined in respect to finding the next decision to be taken.
We have also given semantics of the quantitative level, by showing how this level can be realized in an IDIT, and discussing how this level can be modelled generally.

We have devised a general method for solving IDITs. This method takes outset in a preliminary temporal ordering, and uses this construct as a split tree. A split tree is a tree, in which the nodes are instances of the IDIT being solved. The root of the tree is the original IDIT and the leaves are symmetric IDITs in which all asymmetries, imposed by time variables, are resolved. For each symmetric IDIT we have chosen to use a strong junction tree using lazy propagation for solving them. As a consequence of modelling time variables using χ^2-distributions an exact solution cannot be computed. We have chosen to use a sampling technique, based on the [Gentle, 1998],

for approximating the distributions of the time variables. The combination of this choice and our semantics of time variables has led to the unfortunate event that the solution method results in intractably large functions when solving IDITs with even a small number of time variables.
On a lighter note, we present arguments for the correctness of our algorithm, and have presented other alternatives to our current solution method.

The last part of this conclusion is aimed at the results of this thesis. We present two small examples, which we have solved using our solution method. To give another angle to our solution method we discuss alternative approaches to solving the dilemma we have encountered. These include a nonuniform dynamic discretization and a multi staged Monte Carlo method using local computations. What we find intriguing about these approaches is that the discretization is proposed for use on hybrid Bayesian networks in junction trees, and the multi staged Monte Carlo method was devised for influence diagrams, a framework, which is closely related to IDITs.
Finally, we have discussed the framework of IDITs. In this discussion we present the complications we have found and we propose constructing a number of specialized frameworks as opposed to a single framework for handling all aspects of time, such as IDITs.

Unfortunately, we have not had sufficient time to try out all our proposals, nor to devise an implementation of our framework. We have, however, expanded both DPITs and IDITs, proposed a temporal ordering of IDITs, devised a structure for representing IDITs in a manner, which exposes all asymmetries, and we have devised a general solution method for IDITs.

7.1 Future Work

This thesis documents the study of decision problems involving time, and in the course of this work many interesting aspects of time, and its impact on how a framework should be constructed, have been disclosed. However, we have not had time to explore all of them fully, so we present some of these aspects in the hope that others will continue this work. We divide the aspects into three categories, these being: the aspects relating to the expressive power of IDITs, the aspects relating to the solution method, and the aspects applying the work to real world problems.
During the analysis of several DPITs an element of time, which is not modellable in IDITs, was discovered. Decisions, which must be taken at a specific point in time, which we have opted to call *fixed time decisions*. These cannot be correctly portrayed in IDITs. The problem with fixed time decisions is the semantics of the decision. For instance, if the decision taker, at the time a fixed time decision must be taken, is in the midst of executing the timed action imposed by his choice in another decision. Should the timed action be skipped, and the fixed time decision be taken instead? Or should all decisions before a fixed time decision have a choice, which is resolved

instantly, thus making it possible to choose this choice and in this manner skip to the fixed time decision? Future research should analyze the need of fixed time decisions, find a suitable semantics for them, and, based on the semantics, extend IDITs to be capable of modelling such decisions.

The solution method devised for solving IDITs in this thesis is an general solution method, which, unfortunately, is intractable on models, which handle more than a few time variables. This is because of the exponential nature of the functions we use to represent utility potentials. Therefore, we propose that a method for solving IDITs should be devised, which focuses on keeping the intermediate results and calculations to a minimum with respect to size. Function analysis could be of used to find some regularities in the function expressions and use this to approximate the large expressions by smaller ones.
An approach could be to first alter the semantics of time variables, so they only represent an uncertainty in time, and not a specific point in time. This results in the time variables not being dependent on each other, which is one of the aspects, which lies as the foundation of our problem.
In Chapter 6 we have proposed other approaches to handling the continuous elements of IDITs. An interesting approach to consider is that of a nonuniform dynamic discretization [Kozlov and Koller, 1997], as this approach also uses the confinements of a junction tree to structure the variables of some network.
The approach we discuss in Chapter 6, proposed in [Charnes and Shenoy, 2002], was proposed to handle potentials, which are intractably large. This is exactly the problem we end up having. Unfortunately, this approach is aimed at models in which all variables are discrete, so either a discretization must be performed beforehand, or the approach should be modified to also handle hybrid networks.

The framework of IDITs, as proposed in this thesis, is yet only applicative as a tool for communicating DPITs, as it is not yet implemented. The use of sampling and the complexity of the solution method makes even small IDITs hard to solve by hand. An implementation could take outset in the solution method proposed, yielding it possible to solve small to medium sized IDITs, however, larger IDITs would be too time and space consuming for computers to handle. Assuming that IDITs are implemented considerable speed ups can be achieve by utilizing the equality of the symmetric IDITs. As mentioned in Chapter 5 there exists an exponential amount of symmetric IDITs in the number of split variables. All these symmetric IDITs are converted to a strong junction tree. By studying the symmetric IDITs, however, it is deduced that many of these are equivalent, as splits are also caused by restriction functions, which do not change the structure of the IDITs, but only the state space of the decision variables in it. An implementation would also make it possible to argue about the use of sampling as opposed to, for instance, approximating the probability distribution of time variables using polynomials or any other method of approximation.

Bibliography

[Bielza and Shenoy, 1999] Bielza, C. and Shenoy, P. P. (1999). A Comparison of Graphical Techniques for Asymmetric Decision Problems. *Management Science*.

[Broe et al., 2003] Broe, M., Jeppesen, R., and Nielsen, S. H. (2003). Representing Decision Problems Involving Time. Technical report, Department of Computer Science, Aalborg University.

[Charnes and Shenoy, 2002] Charnes, J. M. and Shenoy, P. P. (2002). A Multi-stage Monte Carlo Method for Solving Influence Diagrams using Local Computations. Working paper, http://lark.cc.ukans.edu/ pshenoy/.

[Cover and Thomas, 1991] Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. John Wiley and Sons.

[Demirer and Shenoy, 2001] Demirer, R. and Shenoy, P. P. (2001). Sequential Valuation Network: A New Graphical Technique for Asymmetric Decision Problems. In *Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 252–265. Springer-Verlag.

[Dittmer and Jensen, 1997] Dittmer, S. L. and Jensen, F. V. (1997). Myopic Value of Information in Influence Diagrams. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 142–149. Morgan Kaufmann Publishers.

[Gentle, 1998] Gentle, J. E. (1998). *Random Number Generation and Monte Carlo Methods*. Springer-Verlag New York, Inc.

[Grimmett and Stirzaker, 1992] Grimmett, G. R. and Stirzaker, D. R. (1992). *Probability and Random Processes 2nd Ed.* Oxford.

[Horvitz and Rutledge, 1991] Horvitz, E. and Rutledge, G. (1991). Time-Dependent Utility and Action Under Uncertainty. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 151–158. Morgan Kaufmann Publishers.

[Horvitz and Seiver, 1997] Horvitz, E. and Seiver, A. (1997). Time-Critical Action: Representation and Application. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 250–257. Morgan Kaufmann Publishers.

[Howard and Matheson, 1981] Howard, R. A. and Matheson, J. E. (1981). Influence Diagrams. *The Principles and Applications of Decision Analysis*, II.

[HUGIN Expert, 2003] HUGIN Expert (2003). Hugin Decision Engine (v6.0). A piece of software. Found through the Internet on *http://www.hugin.com/*.

[Jensen et al., 1994] Jensen, F., Jensen, F. V., and Dittmer, S. L. (1994). From Influence Diagrams to Junction Trees. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 367–373. Morgan Kaufmann Publishers.

[Jensen, 2001] Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer.

[Jensen and Jensen, 1994] Jensen, F. V. and Jensen, F. (1994). Optimal Junction Trees. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 360–366. Morgan Kaufmann Publishers.

[Kjærulff, 1993] Kjærulff, U. (1993). *Aspects of Efficiency Improvement in Bayesian Networks*. PhD thesis, Department of Computer Science, Aalborg University, Denmark.

[Kozlov and Koller, 1997] Kozlov, A. V. and Koller, D. (1997). Nonuniform Dynamic Discretization in Hybrid Networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 314–325. Morgan Kaufmann Publishers.

[Lauritzen, 1996] Lauritzen, S. L. (1996). *Graphical Models*. Oxford Science Publications.

[Lerner et al., 2001] Lerner, U., Segal, E., and Koller, D. (2001). Exact Inference in Networks with Discrete Children of Continuous Parents. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 319–328. Morgan Kaufmann Publishers.

[Madsen and Jensen, 2003] Madsen, A. L. and Jensen, F. (2003). Mixed Influence Diagrams. To appear in ECSQARU 2003.

[Madsen and Jensen, 1999] Madsen, A. L. and Jensen, F. V. (1999). Lazy Evaluation of Symmetric Bayesian Decision Problems. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 382–390. Morgan Kaufmann Publishers.

[Maplesoft, 2002] Maplesoft (2002). Maple 8. A piece of software. Found through the Internet on *http://www.maplesoft.com/*.

[Mathiassen et al., 2001] Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., and Stage, J. (2001). *Objektorienteret Analyse og Design*. Aalborg: Marko.

[MRC Biostatic Unit, Cambridge, UK,] MRC Biostatic Unit, Cambridge, UK. WinBUGS v1.4. A piece of software.

[Nielsen, 2003] Nielsen, S. H. (2003). Influence Diagrams Involving Time. Master's thesis, Aalborg University.

[Nielsen and Jensen, 1999] Nielsen, T. D. and Jensen, F. V. (1999). Welldefined Decision Scenarios. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 502–511. Morgan Kaufmann Publishers.

[Nielsen and Jensen, 2000] Nielsen, T. D. and Jensen, F. V. (2000). Representing and Solving Asymmetric Decision Problems. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 416–426. Morgan Kaufmann Publishers.

[Nielsen and Jensen, 2002] Nielsen, T. D. and Jensen, F. V. (2002). Representing and Solving Asymmetric Decision Problems. Revised Version of [Nielsen and Jensen, 2000].

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

[Raiffa, 1968] Raiffa, H. (1968). *Decision Analysis*. The McGraw-Hill Companies.

[Shachter, 1986] Shachter, R. D. (1986). Evaluating Influence Diagrams. *Operations Research*, 34(6):871–882.

[Shachter and Kenley, 1989] Shachter, R. D. and Kenley, R. C. (1989). Gaussian Influence Diagrams. *Management Science*, 35(5):527–550.

[Shenoy, 1992] Shenoy, P. P. (1992). Valuation-Based Systems for Bayesian Decision Analysis. *Operations Research*, 40(3):463–484.

[von Neumann and Morgenstern, 1944] von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behaviour*. Princeton University Press.

Appendix A

Summary

Decision analysis is a research area focusing on taking decisions in an uncertain context. Trying to formalize decision analysis has resulted in a range of different frameworks, each having their pros and cons. The frameworks typically used are decision trees [von Neumann and Morgenstern, 1944], influence diagrams [Howard and Matheson, 1981], and valuation networks [Shenoy, 1992].
Even though these frameworks have been used to model decision problems, the models are limited to only encompass time on a qualitative level, that is, having a temporal ordering of the events of the decision problem. However, time also has a quantitative level. Decision problems are models of real world problems, but, as decision problems are modelled without a quantitative representation of time, these do not reflect this dimension of the real world. [Horvitz and Rutledge, 1991] and [Horvitz and Seiver, 1997] discuss how time may influence the utility functions of a decision problem. But the influence of time exceeds the influence of utility functions, for instance, time often influences which choices are possible in a decision. As an example, one cannot go to the stadium at ten o'clock to see a football match, which ended at eight o'clock. In [Broe et al., 2003] the class of decision problems involving time was analyzed and a framework, for modelling these, was proposed. The analysis showed how decision problems involving time, combine two well studied aspects of decision problems, these being that the problems tend to be asymmetric and that time should be represented by continuous variables. In this thesis we have continued the work of [Broe et al., 2003].

Decision problems involving time, DPITs, are characterized by the elements of the individual DPIT being influenced by time. We have found that a DPIT can be divided into four parts, these being: the variables in the DPIT, the decision taker, the preferences of the decision taker, and the relations between the variables. The variables represent the decisions, the uncertainty of circumstances, and time in the DPIT. The decision taker is the person to whom the DPIT is presented. Finally, relationships between variables is represented as probabilistic dependencies, precedence of taking one decision before another, or a restriction of the possible choices

of a decision.
The analysis of DPITs has led to a series of requirements for frameworks modelling these. These requirements ensure that the framework: models time as a continuous element, and does so in a manner resulting in both a controllable and an uncontrollable element; is capable of modelling restrictions of state spaces of decisions; can model time being influenced by other variables; can model observations, which can only be taken in specific time spans; can model chance variables influenced by time; and can model the preferences, of the decision taker, being influenced by time. It should also be modellable that some preference of the decision taker is not realized until some time in the future. Furthermore, the precedence of one decision in relation to another decision may dependent on the point in time they are to be taken.

To satisfy the requirements, for a framework modelling DPITs, we have constructed a framework, named influence diagrams involving time, IDITs, which models such decision problems. The framework builds on the ideas of influence diagrams, such that it models DPITs on two levels. On the qualitative level IDITs models variables and local utility functions as nodes in a directed labelled graph, and the relationships between variables as the arcs of the graph. The semantics of an arc differs according to the nodes it connects, and whether or not the arc is labelled with a guard. Likewise, it can be seen directly in the graphical representation, if one variable restricts another. On the quantitative level information, relating to the individual variables, is given. That is, to each variable the state space of the variable is given, and probability distributions are associated to chance variables. Furthermore, all restriction functions are specified on this level and the functions relating to each local utility function.
We describe IDITs by giving an informal introduction to the concepts of IDITs, and their graphical representation. To exemplify this we have proposed a DPIT, which we have named the SAR problem. It takes outset in the rescue mission set in motion when a person is reported missing. The example includes time on a quantitative level, as the success of the mission is related to finding the missing person alive, within some time frame. We have introduced time variables, which are not directly associated with decisions, in order to model local utility functions realized after the last decision. After describing the ideas of IDITs we define the syntax for both the qualitative and the quantitative level of IDITs. We then describe how the IDIT can be read according to a temporal ordering, which is deduced from the two levels. Furthermore, based on this temporal ordering, we argue for IDITs being a welldefined framework, that is, when a temporal ordering of decision variables exists the next decision can be unambiguously identified.

As a modelling tool the representation language of IDITs would be enough. However, we want to find a policy for taking each decision of the IDIT, therefore, a general solution method is needed. We propose a solution method for solving IDITs, with respect to finding an optimal strategy.
The influence of time can be divided into two main categories, namely one relating to time as a continuous element, and one, which renders the models asymmetric. In the solution method we recursively resolve all asymmetries of an IDIT by splitting the

IDIT into symmetric sub-problems. We end up with a number of totally symmetric sub-problems, which we have organized in a tree structure, called a split tree. We then solve the part of each symmetric sub-problem, which is unique to this specific sub-problem, using a solution method inspired by the solution method for influence diagrams, however, the sub-problems are solved in an environment which has continuous variables. We have chosen to use a strong junction tree approach when solving the symmetric sub-problems. When a solution for the unique part of a sub-problem is found, we merge this result with all other results of sub-problems resulting from this split. This is continued until all results have been returned to the root of the split tree, and the original IDITs is then solved. In this manner we end up with an optimal strategy for taking all the decisions in the IDIT, which is the solution we aimed at.
As we have continuous variables we need to approximate the probability distributions of these in order to obtain the solution. The solution method proposes a sampling technique for this approximation. However, we conclude that sampling over the continuous distributions yields expressions, which grow exponentially in the number of time variables, thus is not applicable for large problems.

Finally, we present two examples, which illustrate some of the important parts of IDITs, and we have solved these using the solution method. As a consequence of the exponential growth in function size we have not found a numerical solution to these examples. However, the examples still serve as an illustration of all other aspects of the solution method.
Furthermore, we have discussed different approaches to approximation, such that future research might find a numerical solution to IDITs. Finally, we discuss the appropriateness of the framework proposed, as opposed to having proposed smaller frameworks.