UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Master Thesis
Advanced Artificial Intelligence

Representing Asymmetric Decision Problems with Decision Analysis Networks

Caroline Leonore König

Madrid, 2012

Universidad Nacional De Educación A Distancia
Escuela Técnica Superior De Ingeniería Informática
Department Of Artificial Intelligence

Master Thesis
Advanced Artificial Intelligence

Representing Asymmetric Decision Problems with Decision Analysis Networks

Presented by:
Caroline Leonore König

Supervised by:
Dr. Francisco Javier Díez Vegas
Dr. Manuel Luque Gallego

Madrid, September 2012

Abstract

During the last two decades several specific decision analysis formalisms for the representation of asymmetric decision problems have been proposed as the common decision analysis formalisms, influence diagrams (IDs) and decision trees (DTs) are not able to represent asymmetric decision problems efficiently. Although those formalisms provide different solutions, none of them has been used in practice to represent real-world problems, what might be a sign that they are not simple enough to facilitate the construction of the model or the communication with the expert. The latter is very important in fields such as medicine where the expert needs to understand the system to accept its advice. For these reasons a new probabilistic graphical model, decision analysis networks (DANs) were proposed by Díez & Luque (2010), which intend to represent the asymmetric aspects of decision problems more naturally.

The main contribution of this work is a revision of DANs from the point of view of syntax and semantics regarding the representation of asymmetric aspects and a comparison of the features of DANs to the previous decision analysis formalisms. First this work presents a review of several previous formalisms and a detailed description of the approaches these formalisms take for the representation of order asymmetry and structural asymmetry, illustrating each method with the representation of three typical asymmetric decision problems taken from the literature. Secondly these alternative representations are compared with detail to the DAN representation, what makes the strengths and weaknesses of each formalism evident. This comparison led further to the improvement of the DAN formalism, because some loose ends and ambiguities were detected. After improving DANs with some refined features, DANs compare now equally or even favorably with the other decision analysis formalisms. As a result of the comparison, we confirm that DANs are a suitable decision analysis tool, first because DANs provide a natural representation of both order and structural asymmetry and second because DANs represent problems with local descriptions, which are independent from the complexity of the problem, what makes DANs suitable for the representation of many problems that cannot be represented efficiently with almost all of the alternative formalisms.

Finally another important contribution of this work is the implementation of DANs at OpenMarkov, an open-source software tool for the edition and evaluation of probabilistic graphical models with the objective that DANs can be used in practice for decision analysis.

Resumen

En las últimas décadas se han propuesto varios formalismos para el análisis de decisiones específicos para la representación de problemas asimétricos, dado que los formalismos genéricos, como diagramas de influencia y arboles de decisiones no pueden representar problemas asimétricos eficientemente. Aunque estos formalismos proporcionan diferentes soluciones, ninguno de ellos ha sido utilizado en la práctica para representar problemas reales, lo que puede significar que no son lo suficientemente sencillo para facilitar la construcción del modelo o la comunicación con el experto. Este último aspecto es muy importante en algunos campos, como por ejemplo la medicina, donde el experto necesita entender el sistema para aceptar su consejo. Por estas razones un nuevo modelo probabilista gráfico (MPG), los redes de análisis de decisiones (RADs) han sido propuestos por Díez & Luque (2010) para representar los aspectos asimétricos de problemas de decisión con más naturalidad.

La contribución principal de este trabajo es una revisión de los RADs a nivel sintáctico y semántico en relación con la representación de asimetría y una detallada comparación de los RADs con los formalismos anteriores. Este trabajo presenta primero una revisión de los diferentes formalismos anteriores y una descripción detallada de los métodos que utilizan estos formalismos para representar asimetría estructural y de orden, ilustrando las soluciones con tres problemas asimétricos que son usados en la literatura. A continuación estas soluciones son comparadas con la de los RADs lo que hace las ventajas e limitaciones de los diferentes formalismos visible. Esta comparación ha llevado también a la mejora de algunos aspectos de los RADs, ya que se han detectado algunos cabos sueltos y ambigüedades. Tras adaptar las RADs, el formalismo es ahora equiparable o incluso mejor respecto a otros formalismos de análisis de decisiones por los siguientes motivos: Los RADs usan una representación natural para la asimetría de orden y de la estructural y segundo porque RADs representan los problemas con descripciones locales, que son independientes de la complejidad del problema, lo que les hace apto para la representación de muchos problemas, que no pueden ser representados eficientemente con la gran mayoría de los formalismos alternativos.

Finalmente otra contribución importante de este trabajo ha sido la implementación de RADs en OpenMarkov, un programa libre para la edición y evaluación de MPGs, con el objetivo que RADs puedan ser utilizados en la práctica para el análisis de decisiones.

Acknowledgement

I would like to thank my family for the support they have giving to me for this master studies.

Additionally I would also like to give the thank to the Profs. Francisco Javier Díez Vegas and Manuel Luque Gallego for their support and the time for revising this master thesis.

Contents

List of Abbreviations

AI	artificial intelligence
AID	asymmetric influence diagram
AVN	asymmetric valuation network
BN	bayesian network
CISIAD	centro de Investigacion sobre Sistemas Inteligentes de Ayuda a la Decision
CPT	conditional probability distribution
CVT	compatibility values table
DAG	directed acyclic graph
DAN	decision analysis network
DT	decision tree
extID	extended influence diagram
GT	game tree
ID	influence diagram
PGM	probabilistic graphical model
PID	partial influence diagram
S-DAG	strategy directed acyclic graph
SDD	sequential decision diagram
SID	sequential influence diagram
SVN	sequential valuation network
UID	unconstrained influence diagram
UML	unified modelling language
VN	valuation network

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Motivation

The construction of intelligent systems for decision making under uncertainty is one of the main objectives of Artificial Intelligence (AI) and has been addressed by different paradigms. From the beginnings of AI until the 1990's a great expectation was put on rule-based expert systems, but due to their specificity for a certain domain and their inefficiency for distributed computations at inference they are nowadays less relevant. In contrast, probabilistic graphical models (PGMs), in particular Bayesian networks (BNs) and influence diagrams (IDs), are gaining further importance since their beginnings in the early 1980's. PGMs gave a new approach to the treatment of uncertainty based on the Bayesian probability theory, which allows for efficient inference procedures capable to solve problems with a complexity which could not be addressed by other methods so far. Furthermore PGMs use a graphical model which encodes dependency relationships directly thus providing a qualitative and easy to understand description of decision problems.

Although IDs provide an efficient solution for knowledge representation and reasoning, they have difficulties to represent asymmetric decision problems, i.e., situations where the outcomes of a variable or the decision options are restricted by previous observations and decisions (structural asymmetry) or the order of decisions is undefined (order asymmetry). As real-world problems are more often asymmetric than symmetric, there is a need to find decision analysis formalisms which can represent asymmetry well. Several specific formalisms have been proposed in the last decades, but none of them has been used to build a real-world application, what might be a sign that they are not simple enough to facilitate the construction of the model or the communication with the expert. The latter is very important in fields such as medicine where the expert needs to understand the system to accept the advice of the expert system. For these reasons Díez & Luque (2010) proposed a new formalism, the decision analysis network (DANs), which represents asymmetry more naturally. Until this master thesis, DANs were presented as a new formalism with a theoretical definition of the model and an evaluation method, both illustrated by means of a simple example and compared briefly to the alternative solutions proposed so far.

The main motivation of this work was to extend Díez & Luque (2010)'s research on DANs. First there was a need to carry out a more detailed comparison of DANs to alternative formalisms in order to assure that DANs address all aspects of the representation of asymmetry. In this thesis the modeling capabilities of DANs are analyzed by representing three typical asymmetric decision problems taken from the literature, which contain all known types of asymmetry, and comparing these representation to the solution the alternative formalisms provide. This comparison should make the strengths and weaknesses of the DAN formalism evident and lead to the improvement of the formalism if necessary. Secondly, there was a need to implement DANs in a decision analysis support tool, so that the formalism can find a practical application. We have implemented the DAN formalism in OpenMarkov, an open-source software tool for the edition and evaluation of graphical probabilistic models developed by the CISIAD [1] . This software tool implements already several types of decision analysis formalisms and is available to a wide audience as it is freely available.

[1] CISIAD stands for Research Center on Intelligent Decision-Support Systems a UNED dependent center

1.2 Objectives

Because of the needs described in the previous section, the objectives of this research can be summarized as follows:

1. Analyze the DAN formalism to ensure the correctness and completeness from the point of view of syntax and semantics of the DANs regarding the representation of asymmetry:
   (a) Compare the capabilities of the DAN formalism, regarding the representation of all types of asymmetry, with other decision analysis formalisms.
   (b) Revise the use of the main characteristics of DANs, such as partial temporal order, restrictions and revelation arcs in the representation of decision problems, in order to refine their specification. This task includes the revision of the meaning of restrictions, the scope and conditions under which restrictions and revelation arcs apply and an analysis of the use of revelation arcs to describe information precedence.
2. Implement DANs at OpenMarkov, so that they can be used in practice to represent decision problems.

1.3 Methodology

The character of the research of this thesis is analytic. The objective is to ensure that DANs address all aspects of the representation of asymmetry correctly. The methodology for achieving these objectives is based on an analysis of the capabilities of the DANs for the representation of asymmetry and a critical comparison of their solution to other formalisms. In more detail the methodology is an iterative process, which involves four phases as described in Figure 1.1:

1. Definition of the features of the formalism.
2. Implementation of DANs at OpenMarkov.
3. Representation of typical asymmetric decision problems with DANs.
4. Comparison of the DAN representation of the asymmetric decision problems with other formalism.

Starting from the initial definition of the DAN formalism from Díez & Luque (2010), the first phase is the definition of the characteristics of the DAN formalism. The second phase is the implementation of the network definition in OpenMarkov, which makes it possible the representation of different asymmetric decision problems with DANs in the third phase. In this phase we use some typical asymmetric decision problems presented in the literature. The DAN representation of each decision problem is compared in the following phase to the correspondent solution of each other formalism. This phase includes eventually the revision of solutions already published in the literature. As this comparison may make visible strengths and weaknesses of DANs, the process is restarted again to improve the DAN formalism.

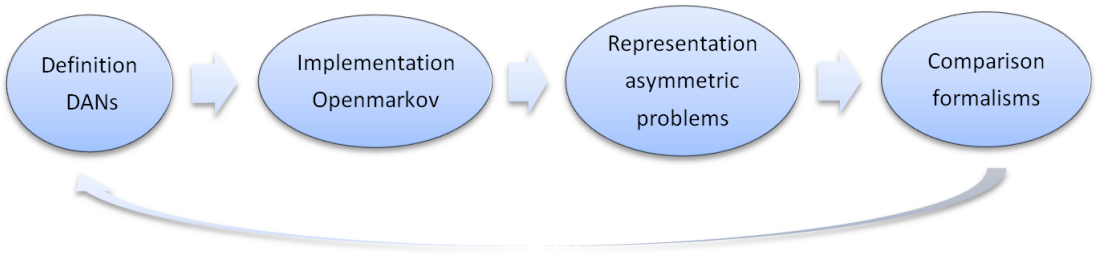


Figure 1.1: Phases in the development of the analysis.

1.4 Organization of the thesis

This master thesis is structured in five chapters. Chapter 1 presents the motivation, objectives and methodology of this research. Chapter 2 reviews the state of the art of decision analysis formalisms for general purpose and specific for the representation of asymmetric decision problems. This part presents an introduction to decision analysis formalisms in general, a description of the fundamentals of PGMs and a discussion of state of several alternative formalisms known so far for the representation of asymmetry. The alternative formalisms examined are decision trees (DT), influence diagrams (IDs), extended influence diagrams (extIDs), sequential valuation networks (SVNs), asymmetric influence diagrams (AIDs), unconstrained influence diagrams (UIDs) and sequential influence diagrams (SIDs). Chapter 3 presents the DAN formalism and explains its capabilities for the representation of asymmetry. The representation of the typical asymmetric decision problem is used to make a comparison for the strengths and weaknesses of the DAN network to the alternative formalisms. Chapter 4 explains the implementation of DANs at OpenMarkov describing the new features of the software system. Chapter 5 presents the conclusions and some open lines for future research.

2 Review of the state of the art

In this chapter we present the basic concepts and the state of the art of probabilistic decision analysis formalisms. First an introduction to decision analysis is made, comprising the description of its origins and its evolution to modern decision analysis and the description of the formal procedure of decision analysis. The second section describes the fundamentals for probabilistic graphical models (PGMs), which are nowadays the most promising model for decision analysis. The third section focuses on decision analysis formalisms regarding the representation of asymmetric decision problems.

2.1 Introduction to decision analysis

2.1.1 Historical background

Every day we face the problem of making decisions under uncertainty, as our perception of the world is based on incomplete information. The necessity of reasoning preceding a decision taking into account the possible factors that may influence the outcome of the decision is recognized since earliest time. The intuitive approach humans take at reasoning is organized in the following steps: (1) foresee of all possibilities that might arise (2) judge how likely each is, based on the perception of the current state of the world and the past experiences and (3) consider the possible consequences (outcomes) of the different acts. This analysis gives us support for making a good decision, at least for simple problems.

Although this kind of reasoning was well known for a long time, it was not until the 17th century that it was formalized with mathematical models based on probability theory. In 1670 Blaise Pascal defined the concept of expected value, which is the weighted average of all possible values of a variable. This concept led to a description of the choice under uncertainty as a rational procedure considering the expected value to be the crucial criterion. The theorem of observed frequencies of Jacob Bernoulli (published eight years after his dead at 1713 in *Ars conjectandi* made it possible to represent mathematically the state of incomplete knowledge or information. Bernoulli stated that when the number of trials is large, the relative frequencies with which things happen will approximate the probability of an event, therefore probabilities are legitimate to predict the occurrence of states of the world. In 1738 Daniel Bernoulli demonstrated by means of the *Sant Petersburg paradox* (Bernoulli, 1954) that the criterion for making a choice should be the expected utility, not the expected value. Bernoulli introduced formally the concepts of utility function and expected utility, which are built upon the personal preferences of a person.

The principles for inductive reasoning are the *Bayes theorem*, named for Thomas Bayes, who formalized the idea of using the inverse probabilities to update beliefs (Bayes & R.Price, 1763). Bayes idea was further extended by Pierre Simon Laplace in his famous work *Theorie analytique des probabilités* (Laplace, 1812). Inference gives insight into a problem as it describes how compelling a piece of information is by assigning it a numerical value. Intuition until then only was able to describe whether a piece of information is relevant for a decision. Laplace applied Bayesian inference to problems in different areas, such as astronomy, medical statistics and even jurisprudence. Although Laplace's application of inference had great success, Bayesian theory was not generally accepted until one century later.

During the first half of the 20ths century several important advances were made in the field of game theory. Wald (1939) renewed some important concepts of statistical analysis as he connected formally classic statistic with decision analysis. He showed that hypothesis testing and parameter estimation are special cases of decision problems. His renewed concepts of statistics were presented more extensively in Wald (1950), which is considered nowadays the paradigm for modern statistics. Another key contribution of this period were the game trees of von Neumann & Morgenstern (1944), which are considered the foundations of game theory. This work formulated the axioms for rationality under which the principle of maximum expected utility hold, and presented a mathematical analysis for strategies for games with imperfect information and multiple players. It was the assertion of the authors that economic behavior problems are identical to the mathematical solutions of strategy games that made this work ground-breaking for decision analysis in any field.

Until the mids of the 20's century decision analysis was based on statistic methods and the interpretation of probabilities was objective, i.e., the probability was understood as the relative frequency of occurrence of an event measured by statistics. Although already von Neumann & Morgenstern (1944) mentioned the idea of subjective probabilities, it was not until Savage (1954) that this idea got foundation. Subjective probabilities represent personal beliefs of an event to happen and are obtained from humans, while objective probability represent the frequency with which events occur and are obtained from statistic data. In consequence objective probabilities are used in machine learning tools, where the plausibility of a proposition is measured, while subjective probabilities are used in decision analysis systems built from the knowledge of human experts. In the mid 1950's two researchers from the Business School of Harvard, R. Schlaifer and H. Raiffa, switched away from the classical decision theory based on statistics and took a different approach of decision analysis in practice. They explained how to apply decision analysis to business by assessing subjective probabilities from experts, and described decision scenarios with decision trees (Raiffa & Schlaifer, 1961). At this period of time also Markovian decision processes (MDPs) were invented, which model decision making in situations where outcomes are partly random and partly under the control of a decision maker. The initial research of Bellman (1957) on MDPs was further extended by Howard (1960), who proposed solution methods based on dynamic programing. R. A. Howard is also known for his contribution of the formal definition of decision analysis from a interdisciplinary point of view (Howard, 1966), which is until now a field of research he is actively working on, and for proposing influence diagrams (Howard & Matheson, 1984).

In the early 1970's arose the idea of using probabilistic inference systems instead of rule-based systems for decision support systems. J. Pearl, a researcher in the field of automated reasoning, made significant research on Bayesian probabilistic models. His work was motivated by the need to study distributed probabilistic computations, which allow to make top-down and bottom-up inferences to overcome the limitations of current rule-based systems, which used an approximate approach for uncertainty management. In the early 1980's, he focused on how a pure Bayesian framework performs at belief propagation and inexact reasoning and demonstrated that Bayesian inference can do deductive, abductive and even intercausal reasoning inference (Kim & Pearl, 1983). He investigated also how directed or undirected graphs can be used as language to encode independence relationships. He recognized the facts that "conditional independence is the most fundamental relation behind the organization of probabilistic knowledge and the most crucial factor facilitating distributed computations" (Pearl, 1993) and that such independencies can be represented suitably with graphical models. Finally at Pearl (1986) he formalized the relationship between graphs and probabilities introducing the d-separation criterion, which allowed to describe conditional independence and which led to belief networks, referred to as Bayesian network (BNs) nowadays.

At almost the same time influence diagrams were presented by Howard & Matheson (1984), which are a graphical representation of the domain knowledge of a decision problem and a alternative to the tree-like representations used so far. The main strength of IDs was their ability to express any relationship between variables with the graphical structure and measure the strength of the relations with probabilities. This approach made IDs a powerful tool for knowledge elicitation at decision problems providing a compact and intuitive to understand model. Nevertheless IDs were criticized initially from different areas of research.

IDs were not accepted immediately by decision analysis researchers, which used so far path diagrams (Wright, 1921) to describe qualitative domain knowledge. The reason they did not accept IDs initially was their aversion to subjective probabilities, which are represented in IDs, because they used models learned from data so far. It was not until one decade later that learning algorithms for IDs were proposed (Spiegelhalter & Lauritzen, 1990; Cooper & Herskovits, 1991). From the automatic reasoning point of view, IDs were also unsatisfactory in its initial form as they were not accompanied with a formal specification, so that a computational tool could not be built. It was the knowledge of the independence relationship descriptions of Bayesian networks (Pearl, 1986) which made it possible to use IDs as decision analysis tools in practice. Some of the evaluation algorithms for IDs (Shachter, 1988; Olmsted, 1983) are based on the transformation of the ID into a decision tree. BNs and IDs are the first types of PGMs.

2.1.2 Definition Decision analysis formalism

Decision analysis addresses the process of decision making in a formal manner. Howard (1966) defined first the formal procedure of decision analysis and did significant research in this field since the early 1960's. In the following, we present some basic concepts of modern decision analysis, which explains the interest for research on decision analysis formalisms and the scope under which we will discuss them in this work.

Howard (1966) defines "Decision analysis [as] a logical procedure for the balancing of the factors that influence a decision. The procedure incorporates uncertainties, values, and preferences in a basic structure that models the decision. The essence of the procedure is the construction of a structural model of the decision in a form suitable for computation and manipulation." This definition summarizes concisely the two main desirable features of any decision analysis formalism: it should allow to create models which represent the structure of the problem and can be treated by computers.

Another interesting concept is the decision analysis process according to Howard (1988), which is depicted in Figure 2.1. The decision analysis process describes how decision analysis is put in practice and which phases it comprises. Starting from a real decision problem the goal of decision analysis is to apply a sequence of steps which provide insight into the problem and allow to understand the choice for the recommended action. This process is formed by the following phases according to Howard (1988):

- *Formulation*: This step fits a real decision problem in a formal model, also termed *decision basis.* The construction of the formal model is termed *elicitation* or *synthesis* and comprises the definition of the decision alternatives, relevant information (relationships or probability assignments that may for example be important in characterizing the connection between decisions and outcomes) and the decision maker's preferences.

- The *evaluation* consists of logical computations, which provide the best decision option.

- The *appraisal* of the analysis is intended to gain insight into why the recommended option is logically correct.

The appraisal may reveal some shortcomings of the analysis, requiring a refinement of the formulation to assure that it is truly appropriate to the problem. At some point, the appraisal step will show that the recommended decision option is right for the decision maker that there is no point in continuing analysis any further.

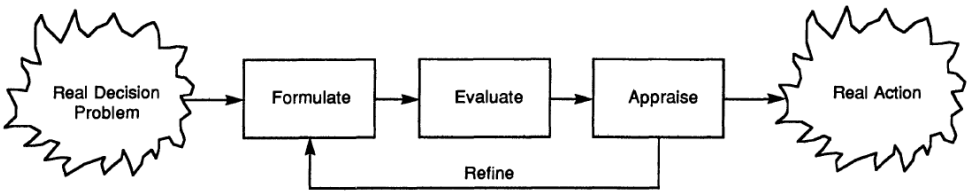


Figure 2.1: Decision analysis process taken from Howard (1988).

This description shows how useful decision analysis, is as it allows to gain insight into a problem and to understand the choice for the recommended action. This process is based on the creation of a formal model of the problem, which explains the motivation for researching on formalisms, i.e., methods and specifications, which can be used systematically to construct a formal model of a decision problem. This model should be a formal description of the problem and allow the decision analyst to understand the structure of the problem and should be treatable by computers in order to analyze the problem efficiently. During the last decades several decision analysis formalisms have been presented. These formalisms usually have graphical models to specify and represent the structure of the problem and different algorithms for inference or learning. Some formalisms are suitable for the representation of symmetric problems, but have difficulties to represent asymmetric decision problems.

The subject of this work is to analyze and compare the capabilities of a formalism for the representation of asymmetric decision problems considering that the resulting model should be clear and descriptive, efficient and treatable by computers. Therefore the scope of this work is the formulation phase. The evaluation and appraisal phase are not explained in detail as it is not the purpose of this thesis to analyze the solution process. Nevertheless the solution methods are briefly described to give a complete view of the formalism and because it is important to consider the representation jointly with the solution method to evaluate the efficiency of the representational model of the formalism.

2.2 Probabilistic graphical models

In this section we introduce *probabilistic graphical models* (PGMs) which are nowadays one of the most important paradigms for reasoning and decision making under uncertainty. A PGM (also termed *probabilistic network*) represents probabilistic knowledge. Many real-world situations can be modeled by a set of entities represented as random variables in a probabilistic network. These models can then form the basis of a decision analysis system to help decision makers identify the most beneficial decision in a given situation.

PGMs use a graph to describe the properties of independence of the joint probability distribution. PGMs have two representational components: the *qualitative* component, which is the graph and the *quantitative* component, which constitute the probabilistic, numerical part. The qualitative component of a probabilistic network encodes visually the dependence and independence assumptions between random variables, whereas its quantitative component specifies the

strengths of dependence relations using probability theory and preference relations using utility theory. While the graph of the qualitative level has expressive power to describe the structure of the problem, which is useful for communication and human knowledge elicitation, the use of a pure probabilistic model at the qualitative level allows for effective (Bayesian) inference. Bayesian networks, for example, can do deductive, abductive and intercausal reasoning.

The main classes of PGMs are *Bayesian networks* (BNs) and *influence diagrams* (IDs). There exist many different variants of PGMs such as temporal models and models specific for the representation of asymmetry, but in this section we focus on the definition of the most basic models, BNs and IDs (which can be considered as extension of BNs). This description of PGMs is partially based on the work of Kjærulff & Madsen (2010).

2.2.1 Bayesian networks

Bayesian networks were proposed by Pearl (1988) and were referred to initially also as belief networks or causal networks. At the qualitative level BNs use a *directed acyclic graph* (DAG) to represent the structure of the domain knowledge. The DAG contains only random variables and the links represent direct dependencies between variables, which are often, but not necessarily, causal relationships. At the quantitative level BNs use a purely probabilistic model formed by conditional probability distributions, which measure the strengths of dependence between variables. The probabilistic model and the uncertainty calculus of BNs are based on Bayesian probability theory.

The DAG of the BN is a description of the joint probability distribution of the probabilistic model in graphical terms. The nodes of the graph represent the domain variables over which the joint probability distribution is defined and the presence and absence of links represent dependence and independence relationship between variables. BNs require a direct correspondence between the probability distribution and the DAG at the graphical level, what means that the topology of the graph must reflect the conditional independence relations of the probability distribution. A formal description of this requirement is based on the concepts of I-map, P-map, Markov conditions and d-separation.

A DAG is said to be an I-map (independence-map) of a probability distribution if all Markov assumptions implied by the graph are satisfied by the probability distribution. A minimal I-map is an I-map, where removing any arc from the graph introduces (conditional) independencies that do not hold in the probability distribution. Conditional independence is described according to the *Markov condition* (or also Markov assumption), which states that a variable X_i is independent from its non descendents ($NonDesc(X_i)$) conditioned on its parents ($pa(X_i)$), what is formally described as:

$$Ind(X_{i;}NonDesc(X_i)|pa(X_i)) \tag{2.1}$$

This definition is an adaption of the causal Markov condition, which states that a phenomena is independent of its non-effects conditional on its direct causes. Nevertheless I-maps do not necessarily reflect all conditional independences of the probability distribution and might not be unique. To overcome this shortcoming the d-separation (directed-separation) criterion was proposed by Pearl (1986), which permits to create P-maps (perfect maps), which capture all conditional independences and are unique. Nevertheless for consistence and evaluation purposes of PGMs the requirement for being the DAG a minimal I-map is enough, meaning that all independence relationships captured at the graph are present at the probability distribution.

The main idea behind considering conditional independence is to take profit of information relevance at the representation of the domain knowledge. Conditional independence permits to "articulate the conditions under which one item of information is considered relevant to another,

given what we already know"(Pearl et al., 1989). Taking into account the conditions under which an information is relevant permits to organize the domain knowledge more efficiently. Based on this assumption PGMs simplify the representation of the joint probability distribution decomposing (or factorizing) it into a product of lower-dimensional conditional probability distributions. Assuming a DAG with a set of variables $\{X_1, ..., X_n\}$, where $pa(X_i)$ denotes a configuration of the parents of X_i, the joint probability over the variables $P(X_1, ..., X_n)$ is factorized as follows:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i)) \tag{2.2}$$

This equation is an adaption of the *Chain rule*, taking conditional independence relations between variables into account, as explained below.

The *conditional probability* of X given Y (denoted as $P(X|Y)$) is the probability of X if Y is known to occur:

$$P(X|Y) = \frac{P(X,Y)}{P(Y)} \tag{2.3}$$

Based on the definition of conditional probability, the *joint probability* of X and Y (denoted as $P(X,Y)$) can be calculated with the following equation:

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{2.4}$$

The independence between variables simplifies the calculation of the conditional probabilities. A variable X is independent of another variable Y with respect to a probability distribution P if the following rule applies:

$$\forall\, x \,\epsilon\, dom(X), \forall\, y \,\epsilon\, dom(Y)\ P(x|y) = P(x) \tag{2.5}$$

Therefore if the variables X and Y are independent, the joint probability can be calculated with the following rule:

$$P(X,Y) = P(X|Y)P(Y) = P(X)P(Y) \tag{2.6}$$

In general, if $X_1, ..., X_n$ are pairwise independent variables, their joint distribution equals the product of marginal probabilities:

$$P(X_1, ..., X_n) = \prod P(X_i) \tag{2.7}$$

A more complex form of independence is the case of *conditional independence*. A variable X is conditionally independent of Y given Z (denoted as $P(x, y|z)$) if the following equation is satisfied:

$$\forall\, x\ , \forall\, y\ , \forall\, z\ .P(x, y|z) = P(x|z) \tag{2.8}$$

In this case the joint probability distribution $P(X, Y, Z)$ can be factorized in the following manner:

$$\begin{aligned} P(X,Y,Z) &= P(X|Y,Z)P(Y,Z) \\ &= P(X|Y,Z)P(Y|Z)P(Z) \\ &= P(X|Z)P(Y|Z)P(Z) \end{aligned} \tag{2.9}$$

The foregoing assumptions leads to the *Chain rule*, which permits the calculation of any value of the joint distribution of a set of random variables using only conditional probabilities.

For a probability distribution, $P(X)$, over a set of variables $X = X1, ..., Xn$, the chain rule decompose it into a product of conditional probability distributions:

$$\begin{aligned} P(X_1, ..., X_n) &= P(X_1|X_2, ..., X_n)P(X_2|X_3, ..., X_n) \\ &= P(X_1|X_2, .., X_n)P(X_2|X_3, .., X_n)...P(X_{n-1}|X_n)P(X_n) \\ &= \prod_{i=1}^{n} P(X_i|X_{i+1}, ..., X_n) \end{aligned} \tag{2.10}$$

Assuming independence relationships between variables allows to simplify the calculation of the joint probability distribution in Equation 2.10, thus leading to Equation 2.2.

2.2.2 Influence diagrams

Influence diagrams can be considered as Bayesian networks augmented with decision variables and utility functions, and provide a representation for decision problems for a single decision maker and a fixed order among the decisions (see Section 2.3.4 for an extensive description). At the graphical level, decisions are represented by rectangular shaped nodes and utility functions as diamond shaped nodes. The links between variables denote direct dependence or informational constraints or describe the domain of a utility function. At the quantitative level the underlying probabilistic model of BNs is augmented with utility functions, which represent the preferences of the decision maker according to utility theory.

Figure 2.3 shows the representation of a simple diagnosis problem by means of an influence diagram, which is based on the BN representation of Figure 2.2. The problem describes the diagnosis for dyspnea. Dyspnea is a symptom of bronchitis and lung cancer. Bronchitis may be caused by a respiratory virus which also causes fever. The doctor can perform a simple test by measuring the temperature in order to decide the treatment for a viral infection or decide for further more specific tests.

The BN describes the causal relations of the problem, while the ID representation describes the decision problem representing the decision to measure the temperature and its cost, the decision for treatment or further tests and the quality of life influenced by the decision of treatment.

2.2.3 Decision making and reasoning

BNs and IDs are applicable in any domain with inherent uncertainty, as they represent in a domain-unspecific way the probabilistic knowledge about the problem. BNs are probabilistic models for *reasoning under uncertainty*, whereas IDs are probabilistic models for *decision making under uncertainty*. Reasoning under uncertainty is the task of computing updated beliefs in (unobserved) events given observations on other events whereas decision making under uncertainty is the task of identifying the (optimal) decision strategy for the decision maker given observations. BN models are the basis of performing Bayesian inference and analyzes about the domain.

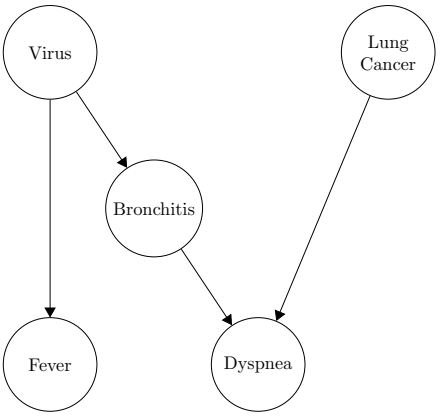


Figure 2.2: BN representation.

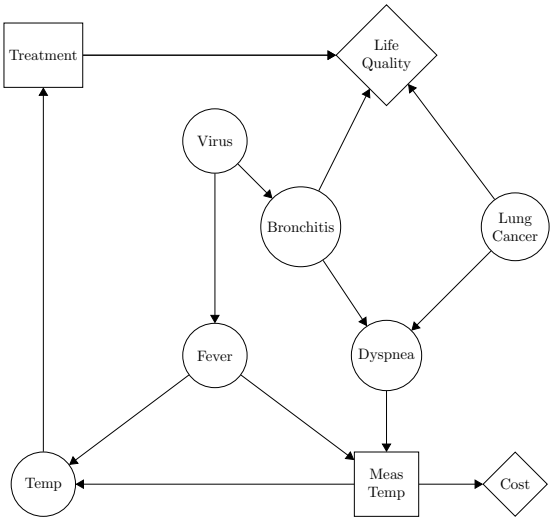


Figure 2.3: ID representation.

2.2.4 Conclusion

We have seen that PGMs are a powerful tool for knowledge elicitation for decision problems, providing a compact and intuitive to understand model with the possibility for inference for decision analysis. DAGs represent the probabilistic model in a compact form as they represent the joint probability function in a factorized form induced from the dependence and independence relations between variables. If independence relationships are not taken into account the size of the probabilistic model grows exponentially with the number of variables, as the joint probability function must contain a probability for each configuration of variables. Therefore DAGs represent efficiently complex systems with a large number of variables, which could not be presented without describing the independence relations between variables.

Another important aspect of graphs is its descriptive power at the qualitative level. As the graph describes visually the relationships between variables, it is an excellent communication tool for formulating, communicating and discussing qualitative interaction models. The description of the causal relations between variables is useful both in problem domains where the causal or correlational mechanisms are (at least partially) known and in problem domains where such relations are unknown, but can be revealed through learning from data.

Further PGMs are a suitable analysis framework, as they are based on a precise mathematical theory which supports reasoning and decision making under uncertainty. PGMs allow to perform deductive, abductive and intercausal reasoning based on the calculus of the Bayes theorem. The ability to perform intercausal reasoning is one of the key differences between automatic reasoning systems based on probabilistic networks and other systems based, for example, on production rules. Another strength of PGMs is their independence from domain specific knowledge as the model is built upon probabilistic relationships between domain variables.

2.3 Representation of asymmetric decision problems

This section describes the state of art of the representation of asymmetric decision problems. The outline of this section is an overview of the different formalisms for the asymmetric decision problems in the first subsection. The second subsection presents a definition of asymmetry followed by a description of three typical asymmetric decision problems from the literature. The following subsections provide a detailed description of the most relevant formalisms and its solution for the different asymmetric decision problems. In particular we analyze the strengths and weaknesses of decision trees (DTs), influence diagrams (IDs), extended influence diagrams (extIDs), sequential valuation networks (SVNs), asymmetric influence diagrams (AIDs), unconstrained influence diagrams (UIDs) and sequential influence diagrams (SIDs).

2.3.1 Overview asymmetric decision analysis formalisms

IDs and DTs are the two formalisms most widely used for the representation of decision problems. IDs provide an efficient representation of the structure of the problem, i.e., the description of the conditional independence assumptions, but there exist problems which require the description of additional information. In particular most real-world problems are asymmetric, i.e., they have conditioned scenarios, where the appearance of a variable depends on previous observations or decisions (structural asymmetry) or scenarios where the order of the decisions varies (order asymmetry). These type of problems can not be represented efficiently with common IDs as this formalism does not describe the asymmetric aspects of a problem. On the other hand DTs address the representation of these aspects with ease as they fully depict the scenarios of a decision problem by showing explicitly the order of decisions and observations. However DTs can not represent medium to big real-world problems as their size grows exponentially on the number of variables.

As IDs were not able to represent asymmetry efficiently, different researchers focused on this issue and several formalisms have been proposed from the early 1990's until now. In chronological order, the first alternative formalisms were extensions of IDs with decision-tree based models to capture the asymmetric aspects of the problem, such as the the decision programming language of Call & Miller (1990), the contingent IDs of Fung & Shachter (1990), the extended IDs with distribution trees of Smith et al. (1993) and the decision graphs of Qi et al. (1994). Extended IDs (extIDs) for example, describe asymmetry by showing the conditioning scenarios of a variable in a tree-like structure. Short time later, Covaliu & Oliver (1995) proposed sequential decision diagrams (SDDs), which combine the description of the uncertainty model of IDs with an explicit description of the sequence of variables in a compact graphical structure, which can be seen as schematic decision tree representation, where each variable appears only once.

Asymmetric valuation networks (AVNs) (Shenoy, 1996) were proposed as solution for the representation of asymmetry based on valuation networks (VN). VNs (Shenoy, 1992) are an alternative to IDs and DTs as they use arbitrary probability valuations for the description of the probabilistic model, which allow to specify the joint probability with flexibility as they do not require a factorization in a conditioned form. Some time later Demirer & Shenoy (2006) proposed sequential valuation networks (SVNs), which combine features from SDDs and AVNs and fix some shortcomings of each of the two formalism (Bielza & Shenoy, 1999), which are mainly the inconsistency of the state space of variables of SDDs, the preprocessing of probabilities required to bring a SDD to a consistent form and the inability of VNs to express the asymmetric structure of the problem at the graphical level. The resulting formalism, the SVN takes advantage of the relaxed requirements of the description of the probability model of VNs and the expressiveness of the description of the sequential and asymmetric aspects of SDDs.

Another proposal are asymmetric influence diagrams (AIDs) (Nielsen & Jensen, 1999a), which adapt the semantics of an ID for the representation of decision problems with conditioned scenarios. The original requirements of IDs enforce a sequential representation of the decisions that is not natural for asymmetric decision problems. AIDs provide a framework where certain decision problems can be represented with a partial order describing under which conditions an arc or a node is possible. Unconstrained influence diagrams (UIDs) (Jensen & Vomlelová, 2002) are a specific solution for the representation of test and diagnosis problems as they are able to represent decisions without a total order. As an improvement of the different formalisms known so far, Jensen et al. (2006) presented the sequential influence diagrams (SIDs), which combine features from AIDs, SVNs and UIDs and provide a solution for the representation of both order asymmetry and structural asymmetry. A recent review of several formalisms can be found in Bielza et al. (2011).

2.3.2 Asymmetric decision problems

As described by Smith et al. (1993) a decision problem is *asymmetric* when a particular act or event leads to very different possibilities and in consequence not all decisions and variables are considered in all circumstances. This type of asymmetry is termed structural asymmetry and appears when the value taken on by a variable restricts the domain of other variables. Another type of asymmetry is the order asymmetry. This happens when several orderings of the decisions or observations are possible in the decision problem.

An alternative description of asymmetry is given by Shenoy (2000),who defines asymmetry by means of the decision tree representation, which shows all decision scenarios explicitly. In a decision tree a path from the root to a leaf node is called a scenario. A decision problem has structural asymmetry if the number of scenarios in a decision tree representation is less than the cardinality of the Cartesian product of the state spaces of all chance and decision variables. Order asymmetry is shown in the decision tree if the decisions appear in a different order in the branches. Thus a decision problem is symmetric only if its decision tree representation contains all variables at every scenario and the variables appear in the same order.

Examples

In the following we describe three asymmetric decision problems: the n-test problem, the reactor problem and the dating problem. These problems show different types of asymmetry and are used subsequently for the illustration of the different formalisms. In particular the n-test problem is used to illustrate order asymmetry and the reactor and dating problem are used to illustrate structural asymmetry.

n-test problem

The n-test problem was first defined by Luque & Díez (2007) and describes situations where n tests can be performed in any order. Each test result reveals with a certain sensitivity and specificity the state of a chance variable. In medical research the n-test pattern is often used to describe diagnosis problems. For instance see the description of the mediastinal staging of non-small cell lung cancer problem of Luque et al. (2009), where several tests can be performed to discover information about the health state of the patient and then a decision about the most suitable treatment is made. For the purpose of illustration we use in this thesis the diabetes problem (Demirer & Shenoy, 2006), which is a specific case of the n-test problem for the diagnosis of diabetes with two tests. Diabetes can be detected with a blood test, which indicates elevated levels of glucose in blood for diabetic patients. An alternative test is the urine test, which

indicates elevated levels of glucose in urine for diabetics. A doctor has to decide whether to treat or not a patient for diabetes. Before the doctor can decide to order a second test (blood or urine) he knows the result of the first test. The tests are not repeated and the doctor can always observe if the patient shows the symptoms of diabetes or not. The variables involved in the representation of the problem are explained in the next section, along with their probabilities.

- Chance variable *Diabetes*:
  Values: present, absent
  The prevalence of diabetes is 7%.
- Chance variable *Symptom*:
  Values: present, absent
  The symptom of diabetes appears with a probability of 85% if the illness is present and with a probability of 0,1% if the illness is absent.
- Decision *Blood Test*:
  Values: test, not test
  The doctor decides whether to order the blood test or not.
- Chance variable *Blood test result*:
  Values: positive, negative
  This test has a sensitivity of 96% and a specificity of 98%.
- Utility *Cost of the blood test*:
  The utility function associated to performing the blood test.
  The blood test has a cost of 50 .
- Decision *Urine Test*:
  Values: test, not test
  The doctor decides whether to order the urine test or not.
- Chance variable *Urine test result*:
  Values: positive, negative
  The urine test has a sensitivity of 97% and a specificity of 99%.
- Utility cost of *Urine Test*:
  The utility function associated to performing the urine test.
  The urine test has a cost of 30.
- Decision *Therapy*:
  Values: treat, not treat
  The doctor decides whether to apply the therapy or not.
- Utility *Quality of life*:
  The utility function associated to a possible diabetes patient receiving treatment.
  The cost of the therapy for a patient, who has diabetes, is 3 if he does not receive treatment and 8 if he receives treatment. If the patient does not have diabetes, the cost is 9 if he receives the therapy and 10 if he does not receive the therapy.

The diabetes problem shows order asymmetry as the order of the tests is not specified and the doctor can decide to arrange the blood test or urine test in any order. The diabetes problem also shows structural asymmetry as the test result is not available if the test is not performed. This means more formally described that the decision to not perform the test restricts the domain of the test result variable.

Reactor problem

The reactor problem was initially described by Covaliu & Oliver (1995), but we use here an adaption proposed by Bielza & Shenoy (1999).

An electric utility firm has to decide (*Build Decision*) whether to build a reactor of advanced design (ba), conventional design (bc), or no reactor (bn) . If the reactor is successful, i.e., there are no accidents. An advanced reactor is more profitable, but it is riskier. Experience indicates that a conventional reactor has probability *0.98* of being successful (cs) and *0.02* of failing (cf). On the other hand, an advanced reactor has probability *0.66* of being successful (as), probability *0.244* of a limited accident (al), and probability *0.096* of a major accident (am) . If the firm builds a conventional reactor, the profits are estimated to be *$8B* if it is a success and $-$*$4B* if it is a failure. If the firm builds an advanced reactor, the profits are *$12B* if it is a success, $-$*$6B* if there is a limited accident, and $-$*$10B* if there is a major accident. The firm's utility function is linear in dollars. Before making a decision to build, the firm has an option to conduct a test or not of the components of the advanced reactor at a cost of *$1B*. The test result can be classified as bad (b), good (g), or excellent (e). If the test is performed, its results are correlated with the success or failure of the advanced reactor. The likelihoods for the test results are as follows: $P(g|as) = 0.182$, $P(e|as) = 0.818$, $P(b|al) = 0.288$, $P(g|al) = 0.565$, $P(e|al) = 0.147$, $P(b|am) = 0.313$, $P(g|am) = 0.437$and $P(e|am) = 0.250$. If the test results are bad, the Nuclear Regulatory Comission will not permit an advanced reactor. The firm needs to decide (*Test decision*) whether to conduct the test (t), or not (nt). If the decision is nt, the test outcome is *no result* (nr).

The reactor problem shows structural asymmetry as the test result influences the options available for the decision of building the reactor. When the test result is bad, the decision to build an advanced reactor is not available. But if the test is not performed, the decision of building a reactor has not any constraint. This problem has another constraint between the components of the advanced reactor and the test result. When the component of the advanced reactor is successful, the test result cannot be bad.

Dating problem

The dating problem was first described by Nielsen & Jensen (2000), but here we explain a slightly different version from Jensen et al. (2006).

Joe needs to decide whether to ask *(Ask?)* Emily for a date for Friday evening. He is not sure if Emily likes him or not (*LikesMe*). If he decides not to ask Emily or if he decides to ask and she turns him down, he will then decide whether to go to the nightclub or watch a movie on TV at home (*NClub?*). Before making this decision, he will then consult the TV guide to see if there are any movies he would like to see (TV). If he decides to go to a nightclub, he will have to pay a cover charge and pay for drinks. His overall nightclub experience (*NCExp*) will depend on whether he meets his friends (*MeetFr*), the quality of life music, etc (*Club*). If Emily accepts (*Accept*), then he will ask her whether she wishes to go to a restaurant or to a movie (*ToDo*); Joe cannot afford to do both. If Emily decides on a movie, Joe will have to decide (*Movie?*) whether to see an action movie he likes or a romantic movie that he does not really care for, but which may put Emily in the right mood (*mMood*) to enhance his post-movie experience with Emily (*mExp*). If Emily decides on a restaurant, he will have to decide (*Rest?*) on whether to select a cheap restaurant or an expensive restaurant. He knows that his choice will have an impact on his wallet and on Emily's mood (*rMood*) that in turn will affect his post-restaurant experience with Emily (*rExp*).

The dating problem shows both types of asymmetry. Structural asymmetry appears as the decision to ask Emily for the date (*Ask?*) and the decision of Emily to accept (*Accept*) leads to

very different situations. If Joe gets the date he will decide on how to organize the date (*ToDO*, *Movie?* or *Restaurant?*), but if he does not get the date he will take decisions about organizing the night going to a nightclub or watching TV ($NClub?, TV$). In each case the decisions and observations of the alternative option are irrelevant. The dating problem contains also several constraints, where the value of a variable restricts the value of another. For example the decision about whether to see a movie or go to a restaurant (*ToDo?*) restricts the values of the subsequent decisions. If the decision is to go to a restaurant, the decision regarding which type of movie to see is irrelevant. The same happens when the decision is to see a movie regarding the choice of restaurant. The dating problem shows also order asymmetry as the order of the observations *Club* and *MeetFriends*, which influence the overall nightclub experience (*NCExp*) is unspecified.

2.3.3 Decision tree representation

Decision trees (DTs) were first proposed by Raiffa & Schlaifer (1961) and are based on *Game trees* (GTs), which were defined earlier by von Neumann & Morgenstern (1944) and studied extensively in the 1950's. According to the generalization of Kuhn (1953), GTs are also known as extensive-form games. GTs arose from the field of game theory for the specification of games, where the choice of decisions, the movements and the payouts for the possible game outcomes are represented by a tree structure. GTs in fact are capable to represent a decision problem, because a decision problem can be viewed as a game with only one player. Shenoy (1998) studied the representation capabilities of GTs for decision problems and attributed GTs some advantages over DTs due to their flexibility at the representation of information constraints, but GTs were not commonly used for decision analysis, as the decision tree formalisms was presented at the early 1960's for this purpose.

Definition

Decision trees use a tree-like graph, which gives an explicit description of the decision scenarios. The graph is constituted by decision nodes (depicted as rectangles), chance nodes (depicted as circles) and end utilities (usually depicted as triangles, but also sometimes as diamonds) and describes explicitly the order in which the decisions are made and the variables are observed. Each branch depicts an available scenario, where a leaf is an end scenario, i.e., where all variables of the domain take a value. The arcs of the branches represent the occurrence of a chance variable conditioned on the past decisions and events. The size of the DT is exponential on the number of variables because of the explicit representation of the scenarios. Olmsted (1983) proposed a technique called *coalescence* which allows to reduce the size of the tree by collapsing identical sub trees.

Evaluation

The optimal strategy of a DT can be found using recursive dynamic programming methods: this method is also referred to as *averaging-out-and-folding-back* (Raiffa, 1968). The dynamic programming approach is quite efficient as it avoids to enumerate all possible strategies.

DT representation of the diabetes problem

Figure 2.4 shows a part of a DT for the diabetes problem. The order asymmetry is apparent as the nodes BT(*Blood Test*) and UT (*Urine Test*) do not appear in the same order in all scenarios. The DT shows the structural asymmetry as the test result node B (Blood Test Result) is not included in the scenarios where the decision of a test is not to perform this test. The DT

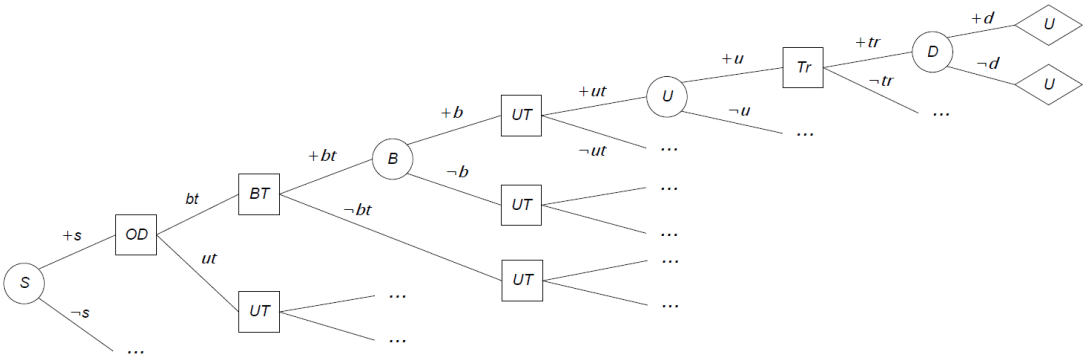


Figure 2.4: DT representation of the diabetes problem taken from Díez & Luque (2010).

representation of the diabetes problem is very large as there are 8 variables each with 2 states, but being some scenarios impossible. The DT has 144 scenarios, what is smaller than if the decision problem was symmetric. In that case the DT would have $2^8 = 256$ scenarios.

DT representation of the dating problem

Figure 2.5 shows the upper branch of the DT representation of the dating problem. The DT has the structural asymmetry because not all variables appear at all scenarios. The decision analyst can easily observe from the model the occurrence of disjoint scenarios. The variables *ToDo*, *Movie*, *Restaurant* are not observed at the scenarios containing the choice of *Ask=no.*The same happens for the variables *NClub*, *TVExp*, *NclubExp*, which are not included in the scenarios where *Accept=yes*. The description of structural constraints of the DT is explicit and very intuitive, but the DT of the dating problem shows also the main drawback of this formalism: the explicit representation of all the decision scenarios makes the model very large.

DT representation of the reactor problem

Figure 2.6 shows the DT representation of the reactor problem. The DT representation of the reactor problem describes the constraint of the test result on *Build decision* explicitly. When the test result is not available the *Result of test* node does not appear in the decision scenario. In this case the *Build decision* has all decision options. If the test result is bad, the *Build decision* node does not show the decision option of build advanced reactor (*ba*). This example shows how the tree describes clearly structural asymmetry as it only depicts possible scenarios. The product of the cardinality of the variables is 108, but there are only 21 possible scenarios. Using coalescence the tree can further reduce its size to only 12 scenarios.

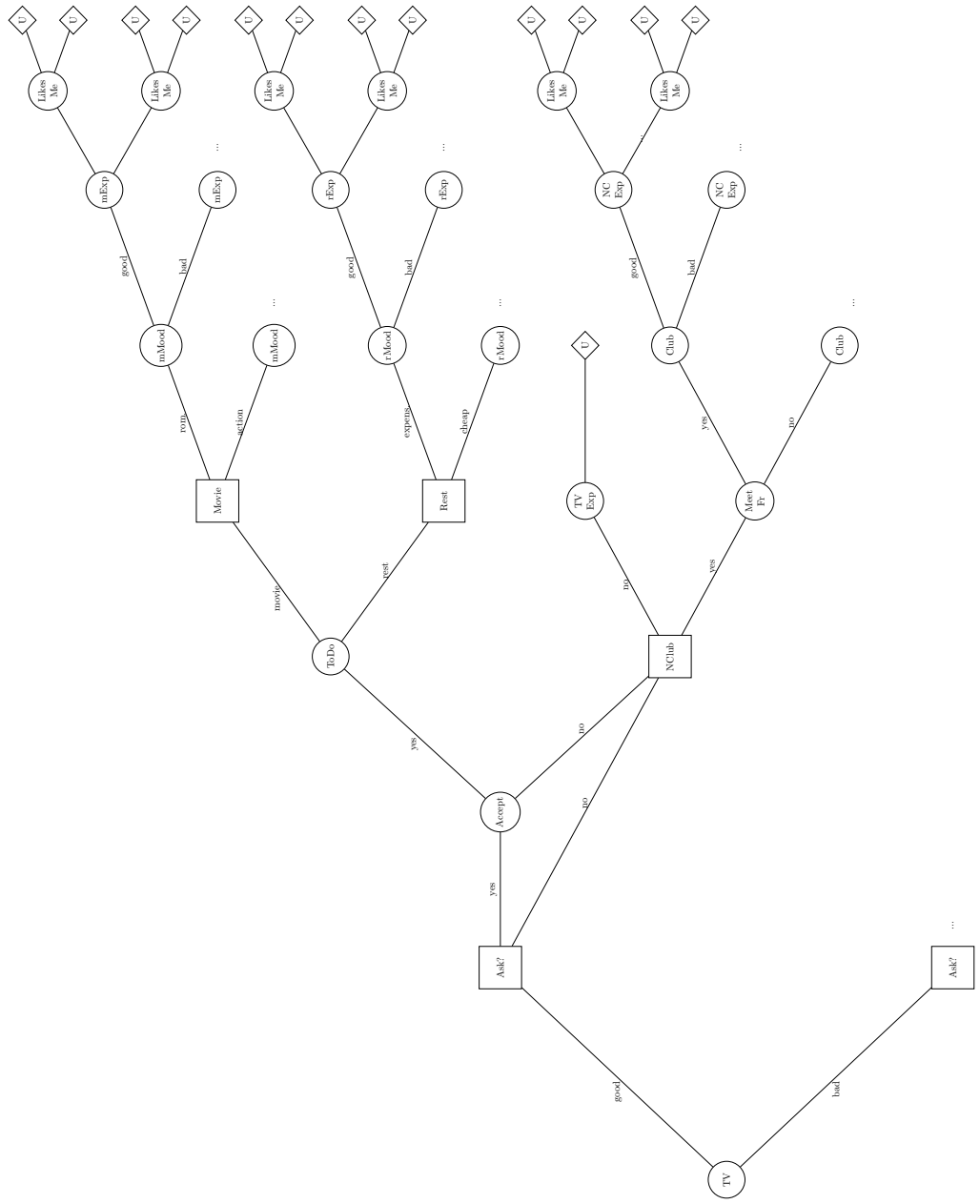


Figure 2.5: A coalesced DT representation of the dating problem.

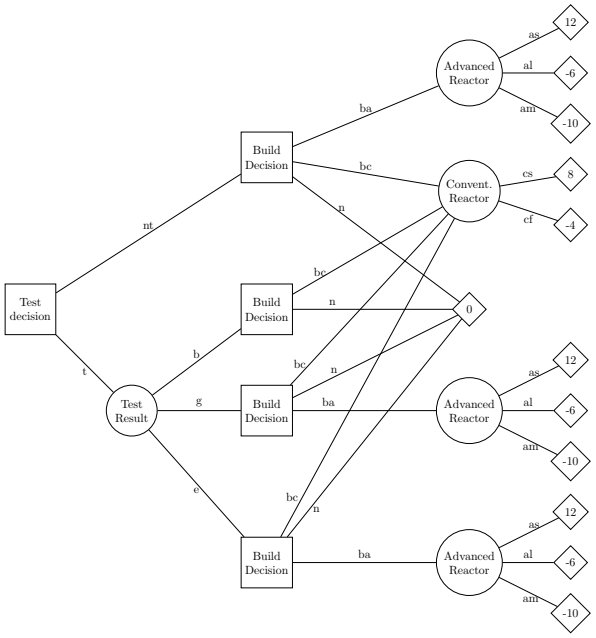


Figure 2.6: A coalesced DT representation of the reactor problem.

The DT representation of the reactor problem appears often in the literature. In fact as almost all authors use it as an example: It was first used at Covaliu & Oliver (1995) to represent the differences to the SDD representation, but it also appears in (Bielza & Shenoy, 1999) and (Bielza et al., 2011) for the comparison of different formalisms. Furthermore this example also appears in (Nielsen, 2001) as part of the comparison to AIDs, in (Demirer & Shenoy, 2006) for the comparison with SVNs and in (Jensen et al., 2006) for comparison with SIDs.

Conclusion

The main advantage of DTs are their simplicity and intuitiveness. As each branch of the tree depicts an available scenario, DTs give a fully detailed view of decision problems, what makes them very easy to understand for decision analysts. But this has also the drawback that DTs can only present decision problems with a small number of variables, as the size of the tree grows exponentially with the number of variables. Nevertheless for small problems, DTs are very suitable. Structural asymmetry is modeled by not representing the scenarios for irrelevant variables, what avoids unnecessary computations. Order asymmetry is modeled by adding a node *Order of decision*, which has a branch for each possible ordering.

Although DTs have absolute flexibility to model asymmetric decision problems, they do not clearly show the dependence and independence relationships among variables in the decision problems. Another limitation is that it requires the probability distribution for each variable conditioned on the past decisions and events and these probabilities may not be the same as those assessed from the decision maker. Decision analysts prefer to assess the probabilities in a causal direction of the variables. Thus a preprocessing of probabilities is necessary to compute the probabilities required in the tree, what is a computational expensive task. Another weakness of DTs is that its model is not easily adaptable to changes in the decision problem. This happens first because the representation is not separated clearly from the solution and next because the probabilities are dependent on the variables which appear at its left. Further details regarding

the expressiveness of the graphical representation of DT for symmetric and asymmetric decision problems can be found in (Shenoy, 1994) and (Call & Miller, 1990).

The representation of the three asymmetric decision problems has shown how the explicit representation of the decision scenarios permits to describe asymmetry. At the representation of the diabetes problem we have seen that DT model order asymmetry by describing with separate branches each possible sequence of the decisions.The representation of the dating and reactor problem has shown that structural asymmetry is modeled by not representing impossible scenarios for irrelevant variables, what avoids unnecessary computations. Nevertheless the main drawback of DTs became also evident. The explicit representation of decision scenarios makes DTs grow exponentially on the number of variables, what makes them unsuitable for the representation of medium to large real-world problems. Only the representation of the reactor problem remained simple because it is a very small problem.

2.3.4 Influence diagram representation

Influence Diagrams (IDs) (Howard & Matheson, 1984) are one of the most common probabilistic networks for decision problems. IDs were presented primarily as front-end of DTs to simplify modeling and analysis of DTs, in particular to avoid any preprocessing of probabilities. Influence diagrams are popular as they are compact and, unlike DTs clearly indicate the dependence and independence assumptions in the model. The initial definition of IDs did not provide a concise definition, so that IDs could not be constructed systematically and used for computations, i.e., it was not possible to use IDs in practice as decision analysis tool. The formal description of the conditional independence relations by the d-separation criterion (Pearl et al., 1989; Pearl, 1986) gave the ID formalism the theoretical foundation for representing probabilistic knowledge with graphs. The d-separation rule is a simple graphical test for detecting the conditioned independence relations implied by the topology of a graph implemented for example at the *Bayes-ball* algorithm of Shachter (1998). This contributions made it possible to create a sound definition of the representation of probability models as DAGs (minimal I-map or P-map), to describe algorithms for their systematical construction and to characterize the set of legitimate graphical transformations (e.g. arc reversal, node removals, etc.). These contributions led to the first formal description of IDs by Shachter (1986), where an ID is defined as a BN augmented with decision and with a single utility node capable to represent and solve a sequential decision problem for a single decision maker assuming the non-forgetting condition. In the following more specific versions of IDs were proposed such as for example the stepwise solvable ID of Zhang et al. (1994), the extended ID with super value nodes of Tatman & Shachter (1990) or the partial ID of Nielsen & Jensen (1999b).

Definition

An ID is basically a BN (see Section 2.2.1 for a detailed description) augmented with decision nodes, which represent decision alternatives and utility nodes, which assign a utility value to each state of the world. IDs represent a single decision maker's beliefs and preferences about a sequence of (ordered) decisions made under uncertainty. The following formal description of IDs is based on the description of IDs of Zhang et al. (1994) and of Nielsen & Jensen, 1999b.

An influence diagram I is defined formally as a quadruple$I = (X, G, P, U)$ as follows:

- $G$is a DAG (X, A) with the node set X and the arc set A, where the set X is constituted by the set of *chance nodes* C, the set of *decision nodes* D and the set of *utility nodes* V**:**
  - *Chance nodes* (drawn as circles) represent events that cannot be controlled by the decision maker.

  - *Decision nodes* (drawn as rectangles) represent actions that can be controlled by the decision maker..
  - *Utility nodes* (drawn as diamonds) represent the preferences of the decision maker that can be for example a cost or a benefit estimation. Utility nodes can not have children.

  Each decision node or chance node has a set, called the frame, associated with it. The frame of a node consists of all the possible outcomes of the (decision or chance) variable denoted by the node. For any node $x \epsilon X = \{X_1, ..., X_n\}$, we use $pa(x)$ to denote the parent set of the node x in the graph and use Ω_x to denote the frame of the node x.
  The set of arcs A have a different meaning based on the type of node they point to:

  - An arc into a chance node describes *probabilistic dependence*. These arcs describe on which variables the conditional assignment of the chance node is conditioned and are also referred to as *dependency arcs*.
  - An arc into a utility node describes *functional dependence*. An arc to an utility node defines the domain of the utility function of the node.
  - An arc into a decision node describes *temporal precedence*. This means the preceding variable or decision is known when the decision of the second node is made (i.e., it is a direct informational predecessor). These arcs are also named *information arcs*.

- P is a set of conditional probability distributions containing one distribution $P(c|pa(c))$ for each chance variable $c \epsilon C = \{C_1, ..., C_n\}$. For each parent configuration $y \epsilon \Omega_{pa(c)}$and each configuration of the chance variable $x \epsilon \Omega_c$, the distribution specifies the conditional probability of the event $c = x$ given $pa(c) = y$.

- U is a set of local utility functions containing one utility function $u(pa(v))$ for each utility node $v \epsilon V = \{V_1, ..., V_n\}$, which assigns a real value to each parent configuration $y \epsilon \Omega_{pa(v)}$. The local utility functions represent additive contributions to the total utility function $U(X)$.

- For a decision node $d \epsilon D = \{D_1, ..., D_n\}$, a value $y \epsilon \Omega_{pa(d)}$is called an information state of d, and a mapping $\delta : \Omega_{pa(d)} \rightarrow \Omega_d$ is called a decision function for d. The set of all the decision functions for d, denoted by $\triangle_d$, is called the decision function space for d.

IDs must satisfy the following two conditions in order to be an unambiguous representation of a single decision maker's view of the world (i.e. an *proper* ID) so that the representation can be evaluated directly by an algorithm:

- The *no-forgetting* condition states that the decision maker remembers all previous decisions and observations (also referred to as *perfect recall*). This implies that the ID does not need to specify any redundant no-forgetting arc, i.e., a chance node can be an immediate predecessor of at most one decision node.

- The *single decision maker* condition states that there must be a directed path which contains all of the decision nodes (also referred to as *regularity* constraint). This requires a *total order* among the decision nodes and implies that the ID has to model the order of decision if the order is unspecified.

The analysis and evaluation of an ID is based on Bayes inference as explained in Section 2.2.3, where further information states and decision policies are taken into account. The next section describes the different algorithms proposed in the literature for the evaluation of an ID.

Evaluation

The first evaluation method was proposed by Howard & Matheson (1984). This method transforms the ID to an equivalent DT and then computes the optimal policy from the DT. Once the graphical representation of the probabilistic model was defined formally by the description of conditional independence (Pearl, 1986; Pearl et al., 1989), alternative algorithms were proposed which evaluate the ID directly without a secondary representation:

- Olmsted (1983) and Shachter (1986) defined several graphical/numerical operations, called reductions which can be used to transform the ID. The evaluation of the ID consists of the sequential reduction of all its nodes using so called value-preserving reductions. These transformations do not modify the maximum expected value and the optimal strategy and comprise operations such as node removals and arc reversals. These algorithms were designed for the evaluation of an ID with one utility node.

- Another evaluation method is the transformation of the ID into a BN (Shachter, 1988; Shachter & Peot, 1992) by transforming decision and value nodes to probabilistic nodes and calculating the optimal decision policy using probabilistic inference algorithms at the BN. Inference on BN was well studied by Pearl (1988) and later improved with more efficient algorithms based on message passing (Lauritzen & Spiegelhalter, 1988; Jensen et al., 1990; Cooper, 1988).

- Tatman & Shachter (1990) proposed a dynamic programming algorithm for an efficient evaluation for IDs with super value nodes, i.e., where the value function can be decomposed into sums and products.Luque & Díez, 2010 proposed an alternative evaluation algorithm for this type of IDs based on variable elimination which outperforms the initial dynamic programming approach.

ID representation of the diabetes problem

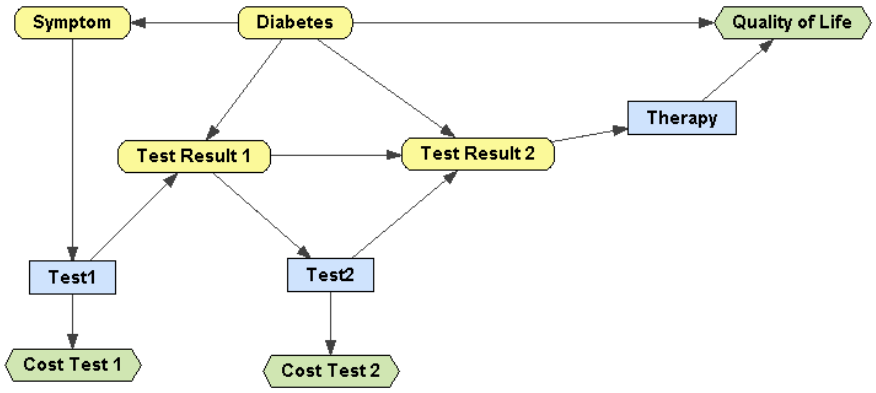


Figure 2.7: ID representation of the diabetes problem.

The ID of Figure 2.7 represents the diabetes problem according to Jensen et al. (2006). The ID representation has problems in representing order asymmetry as they have a requirement that the decisions should be completely ordered. As the order of the tests is not completely defined, it is

necessary to include all admissible decision/observation sequences directly in the representation model. The ID representation contains two additional nodes, which model the choice of the tests. These nodes are the decision nodes *Test 1* and *Test 2*, which model the decision for doing either the blood test, the urine test or not performing any test. The state space of the test result nodes is augmented as it is necessary to add extra (dummy) states which make the problem symmetric. Further the test result has to contain all possible test outcomes from the two tests. In this case the state space of the *Test Result* variable has a cardinality of five, having the states Blood positive, Blood negative, Urine positive, Urine negative and no-result. Figure 2.8 shows the CPT for the *Test Result 1* node with 30 values.

Test1	No Test	No Test	Urine Test	Urine Test	Blood Test	Blood Test
Diabetes	absent	present	absent	present	absent	present
Blood positive	0.0	0.0	0.0	0.0	0.02	0.96
Blood negative	0.0	0.0	0.0	0.0	0.98	0.04
Urine positive	0.0	0.0	0.01	0.97	0.0	0.0
Urine negative	0.0	0.0	0.99	0.03	0.0	0.0
no-result	1.0	1.0	0.0	0.0	0.0	0.0

Figure 2.8: CPT of the variable *Test Result 1.*

The diagnosis problem represented here has an additional requirement, namely that the same test will give the same outcome. For this reason the node *Test Result 1* has a probabilistic influence on the node *Test result 2* which is represented with an additional arc between *Test Result 1* and *Test Result 2*. The conditioned probability distribution (CPT) of *Test result 2* has 150 values being the majority of them impossible, which is described by assigning the zero probabilities to them. Figure 2.9 shows the CPT of *Test Result 2* in a compact form using the tree representation.

The ID representation of the diabetes problem has shown that IDs are unsuitable for representing order asymmetry as they introduce artificial variables which model the sequence of decisions. This example has also shown how inefficient IDs handle structural asymmetry as the use of dummy states increases the space and time requirement for the representation and solution of the problem.

The ID representation of the diabetes problem is large for a two test problem and makes evident that the representation of the n-test problem is infeasible with IDs. The n-test problem is not easily representable with IDs basically due to its inability to model order asymmetry. An ID would need n artificial test decision nodes, each test node having $n+1$ decision alternatives. The correspondent test results would include all possible test outcomes of each of the n tests and a dummy state. So the test result nodes would have at least $2n+1$ states. If furthermore a dependence between the test result is assumed, the probability potentials of the test result nodes are further increased.

At the literature appears another ID representation of the diabetes problem. Bielza et al. (2011)describe an ID representation of the diabetes problem also with the possibility to repeat the test but using auxiliary variables which mediate between the disease and the test result. The use of these mediating variables permits that repeating the test provides the same result and avoids the direct influence of the first test result on the second test result. This solution provides a state space of 60 for either the first and the second test result, while the solution presented here required 30 and 150 states respectively. Nevertheless both solutions show how inefficient the ID representation is even for a two test problem, making it evident that the representation of n-test problems with IDs is unfeasible.

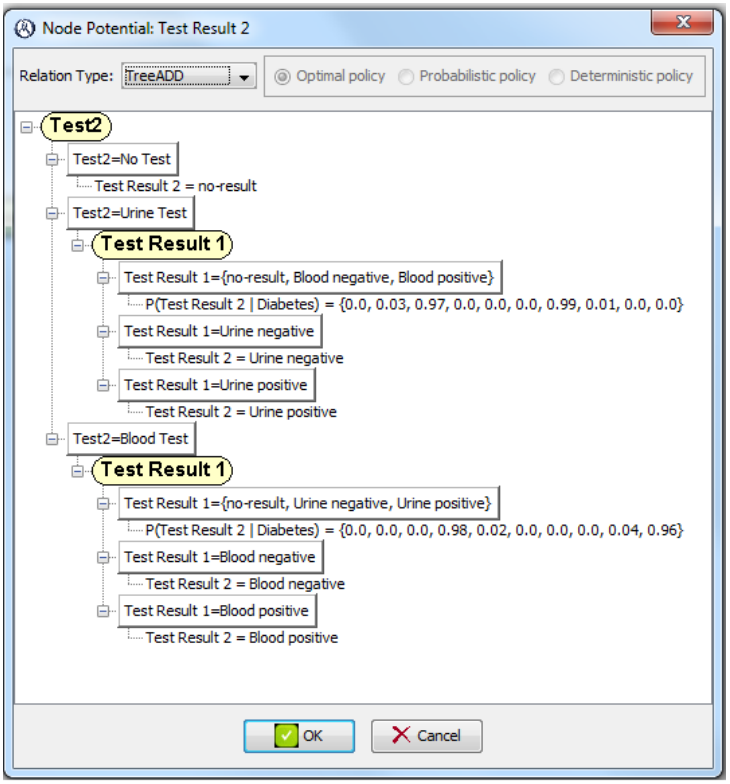


Figure 2.9: CPT of the variable *Test Result 2*.

ID representation of the dating problem

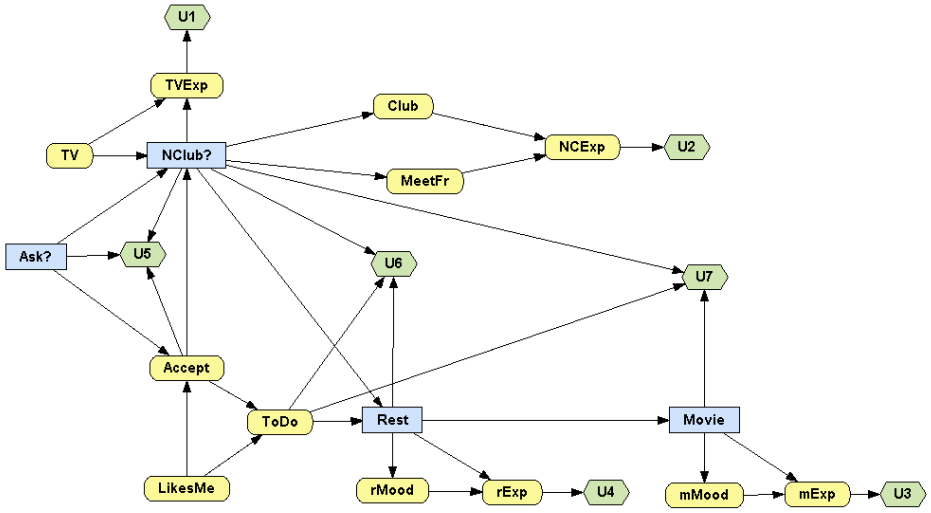


Figure 2.10: ID representation of the dating problem.

Figure 2.10 shows the ID representation of the dating problem. A similar representation of the dating problem as ID appears in Nielsen (2001), although this representation corresponds to the original version of the dating problem (which is slightly different from that we use here) and represents the problem with a partial order. The presented solution here describes the view of a single decision maker for the problem (i.e., it is a proper representation). The representation satisfies the single decision maker's condition as there is a directed path which contains all of the decision nodes, although the order of the decisions *NClub*, *Restaurant* and *Movie* is defined artificially. The requirement of a total order of decisions allows that the problem can be evaluated directly with an reduction-based algorithm but has the inconvenient that the disjoint nature of the decision scenarios is not clear at the graphical level as the decisions seems to happen sequentially. The *non-forgetting assumption* implies that previous decisions and observations are remembered at subsequent decisions. For this reason the ID of the dating problem shows not any redundant non-forgetting arc and each chance variable is at most direct information predecessor of one decision. For example the information precedence of *ToDo* for *Movie* is not expressed explicitly as the value of *ToDo* is known for the prior decision *Movie*. This implicit description of information precedence makes the representation of information constraints more difficult, i.e., the fact that the a decision option is restricted by a variable or previous decision is not obvious from the diagram.

The diagram of the ID reveals no information about the conditioned and disjoint scenarios of the dating problem and its constraint. For example the decision analyst has no information that the decision of *Restaurant* can not appear in the same scenario as *Movie*. The representation of asymmetric constraints is carried out only at the quantitative level. IDs represent structural asymmetry at the quantitative level encoding constraints by an artificial symmetrization of the state space of the variables introducing dummy states, which represent that a variable is impossible in certain scenarios. This artificial symmetrization increases the state space required for the representation of the problem and the computational effort to solve it. Table 2.4 describes the augmented state space of the variables due to dummy states. For each variable appears

a description of the real states and the correspondent dummy state. Table 3.2 shows the size of the potentials (probability potentials, decision function state spaces and utility functions) of each node taking into account the augmented state space of the variable and the number of predecessors.

Restrictions on the outcomes of chance variables imposed by conditioning states are described at the level of the conditioned probability distributions assigning degenerated probabilities. For example the Table 2.1 shows how the outcomes of the variable *TVExp* depend on the conditioning states. The conditioning state *NClub=yes* (Joe decides to go to the club) makes the outcomes of the variable *TVExp* impossible. For this conditioning state zero probabilities are assigned to the real states of the variable and the dummy state *unknown* takes the probability one. Also when the conditioning state is impossible *(NClub=no decision)* the only possible state is the dummy state.

NClub	yes	yes	no	no	no decision	no decision
TV	good	bad	good	bad	good	bad
yes	0	0	0.7	0.5	0	0
no	0	0	0.3	0.5	0	0
unknown	1	1	0	0	1	1

Table 2.1: CPT of the variable *TVExp*.

Restrictions on the legitimate decision alternatives imposed by different information states are not that straightforward to describe as in the case of chance variables. Certain information states can make that the legitimate decision alternatives of a decision are restricted. For example the decision alternatives of the choice of *Restaurant* are restricted by the value of the previous decision *NClub* and the variable *ToDo*. The Table 2.3 for example shows the available decision alternatives for the choice of restaurant for each of the informational states, where the value one denotes that the correspondent decision alternative is available and zero denotes that the decision alternative is not available. As IDs do not implement an explicit control of the legitimate decision alternatives given the information states, IDs model this constraint be assigning large negative utility values to the unavailable alternatives so that these alternatives are not found to be optimal by the solution algorithm.

At the dating problem the decision alternatives of *NClub*, *Restaurant* and *Movie* are restricted. The Tables 2.2 and 2.3 show the available decision alternatives for each informational state, where the value 1 means that the correspondent decision option is available. Impossible decision options are marked with the value 0. Figure 2.11 shows the assignment of the large negative values to the utility nodes *U5* and *U6* in order to express the restrictions on the legitimate decision alternatives.

The representation of the restrictions on decision via the assignment of degenerated utility values is an approach which obscures the representation of the structure of the problem. In principle the nodes $U5$, $U6$ and $U7$ are designated to describe the cost of each real decision alternative. So for example the cost of going to a cheap restaurant is 5 and the cost of going to an expensive is 10 (see Figure 2.11). As the utility nodes associated to the decisions are further used to describe the admissible decision alternatives for certain informational states, the variables which define a restriction are included in the domain of the utility node. For this reason the utility nodes $U5$, $U6$ and $U7$ have additional variables in its domain.

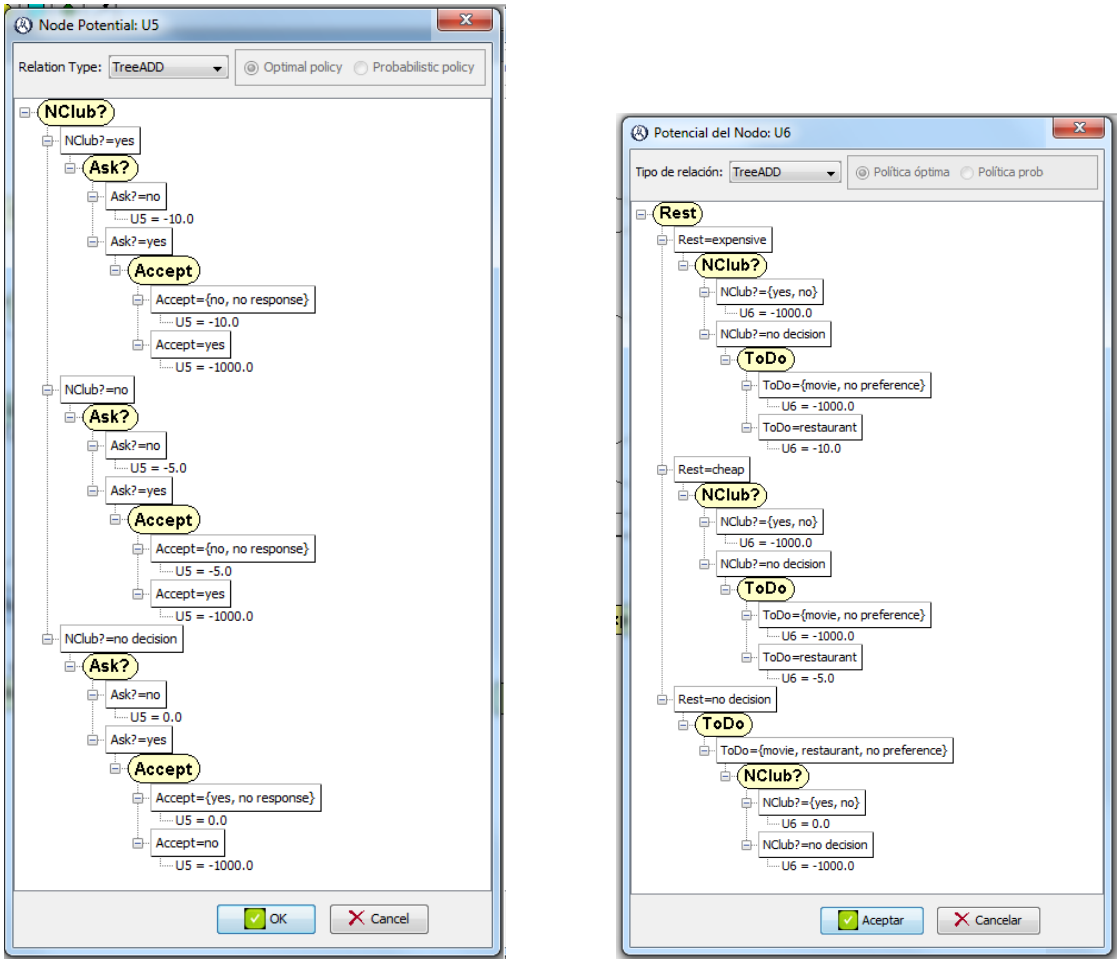


Figure 2.11: Utility values of the node *U5* and *U6*.

Ask	yes	yes	yes	no	no	no
Accept	yes	no	no response	yes	no	no response
yes	0	1	1	1	1	1
no	0	1	1	1	1	1
no decision	1	0	0	0	0	0

Table 2.2: Available decision options for the decision *NClub*.

NClub	yes	yes	yes	no	no	no	no dec.	no dec.	no dec.
ToDo	movie	rest	no pref.	mov	rest	no pref.	movie	rest	no pref.
cheap	0	0	0	0	0	0	0	1	0
expensive	0	0	0	0	0	0	0	1	0
no decision	1	1	1	1	1	1	1	0	1

Table 2.3: Available decision options for the decision *Restaurant*.

Node	Real	Dummy	Size	Node	Real	Dummy	Size
Ask?	yes,no		2	rExp	good,bad	unknown	3
LikesMe	yes,no		2	mExp	good,bad	unknown	3
Accept	yes, no	no response	3	NClub?	yes, no	no decision	3
ToDo	movie, rest.	no preferencce	3	TVExp	good, bad	unknown	3
Restaurant	cheap, exp.	no decision	3	TV	good, bad		2
Movie	rom., action	no decision	3	Club	good, bad	unknown	3
rMood	good, bad	unknown	3	MeetFr	yes, no	unknown	3
mMood	good, bad	unknown	3	NCExp	good, bad	unknown	3

Table 2.4: Size of state spaces of the variables.

Node	Predecessor ID	Potential	Node	Predecessor ID	Potential
Ask?		2	TV		2
LikesMe		2	Club	NClub?	9
Accept	Ask?, LikesMe	12	MeetFriend	NClub?	9
ToDo	Ask?, Accept, LikesMe	36	NCExp	Club,MeetFr	27
Restaurant	NClub,ToDo	27	rExp	rMood,Restaurant	27
Movie	Rest	9	mExp	mMood,Movie	27
rMood	Restaurant	9	U_1	TvExp	3
mMood	Movie	9	U_2	NCExp	3
rExp	rMood,Restaurant	27	U_3	mExp	3
mExp	mMood,Movie	27	U_4	rExp	3
NClub?	Ask?, Accept, TV	36	U_5	NClub,Ask,Accept	18
TVExp	NClub?,TV	18	U_6	Rest,NClub,ToDo	27
			U_7	Movie,NClub,ToDo	27

Table 2.5: Size of the potential of a node.

ID representation of the reactor problem

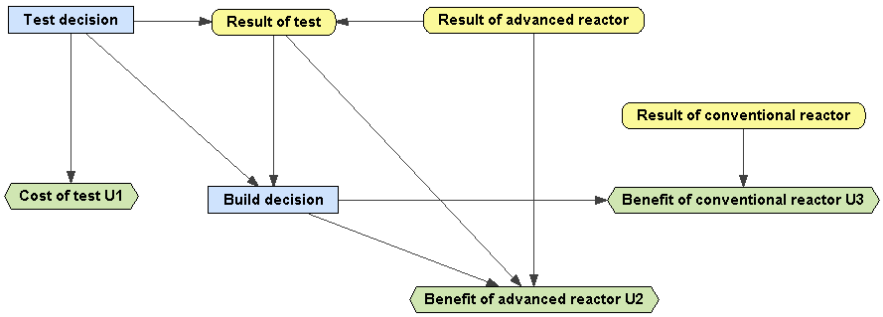


Figure 2.12: ID representation of the reactor problem.

Figure 2.12 shows the ID representation of the reactor problem. The asymmetric constraints of the problem are specified at the quantitative level including dummy states and assigning degenerated probability distributions. For example the constraint that the test result is not available when the decision is not to do the test restricts the outcomes of the test result variable. Figure 2.13 shows the CPT where degenerated probabilities are assigned to the states to describe that the outcome of the test depends on the test decision.

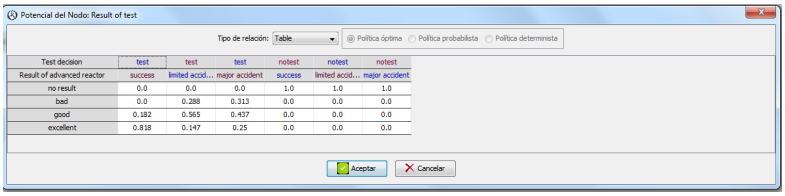


Figure 2.13: CPT of the variable *Result of test.*

The reactor problem contains also a constraint on a decision. The test alternative for building an advanced reactor is not available when the test result was bad. As IDs can not model explicitly the restriction of the decision alternatives for different informational states, they use degenerated utility functions to describe that a decision alternative is not available in certain conditions so that the solution algorithm does not select these decision alternative when computing the optimal strategy. This approach associates large negative values to the informational states which make a decision alternative impossible. This implies that a utility node must be associated with the test decision and that the utility node has at least the variables which define the constraints in its domain. In the case of the reactor problem the informational states are defined by the *Test Decision* and the *Result of test.* As the restriction is only based on the *Result of test* it is not necessary to include the *Test decision* into the domain of the utility node. Further we use the utility node *Benefit of advanced reactor*, which is already associated to the *Build decision* to model the constraint. Therefore the utility node represented in Figure 2.14 contains a large negative value for all combinations which have the conditioning state *Result of test=bad.*

The utility values of Figure 2.14 show also the description of the structural constraints of the *Build decision* and the *Benefit of advanced reactor.* As the benefit of the advanced reactor is only possible when the decision was to build this type of reactor, all utility values corresponding to

the alternative decisions have large negative values to describe that they are not possible states of the world.

The representation of the reactor problem has shown that the solution for representing asymmetry with IDs is an artificial symmetrization. They represent structural constraints by adding dummy states and assigning degenerated probability functions to conditioning states to describe that the outcomes of a variable are restricted. Further they assign large negative values to utility nodes to describe that the decision alternatives are restricted by certain informational states. The use of utility functions to describe constraints on test decisions can obscure the description of the structure of the problem at the diagram as the informational predecessors which define a restriction on a decision are linked to the utility node. For example the utility node of the *Benefit of advanced reactor* only is concerned with the value of the outcome of building an advanced reactor but in order to describe the constraint on the test decision they have also the variable *Result of test* in its domain. Of course modeling the constraint with an additional utility node associated to the *Build decision* would be an alternative. Another drawback of IDs is that they do not show the occurrence of conditioned scenarios at the graphical level. The decision maker gets no information about the constraint of the *Result of test* on the *Build decision* as constraints are modeled at the quantitative level.

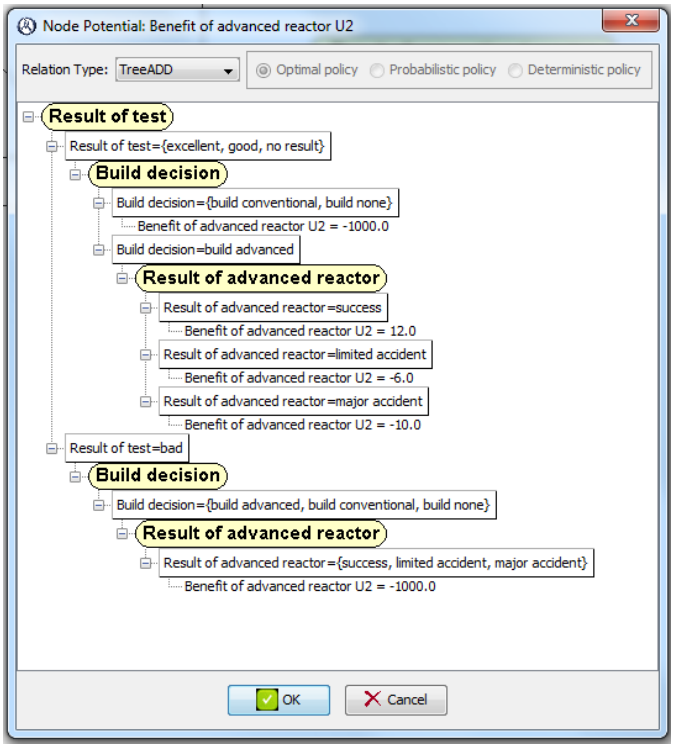


Figure 2.14: Utility values of the variable *Benefit of the advanced reactor.*

The ID representation of the reactor problem appears also in (Bielza & Shenoy, 1999) and Bielza et al. (2011) for the comparison of several formalisms and at Demirer & Shenoy (2006) as part of the SDD representation of the reactor problem.

Conclusion

IDs are a powerful tool for communication, elicitation and detailed representation of human knowledge as they are able to represent clearly the structure of a decision problem. An ID describes the probabilistic knowledge about the problem with chance variables and dependency arcs. Further it represent the actions of the decision maker with decision alternatives and the knowledge about the state of world with information states (described with informational arcs) and the preferences of the decision maker assigning utility values to each state of the world.

IDs are in principle suitable for the representation of sequential decision problems for a single decision maker with perfect recall. As the description of the structure of the problem is modular the ID is easily adaptable to changes at the decision problem and is very compact, i.e., the size of the model is linear on the number of variables. IDs are suitable for the representation of decision problems with conditional probabilities model as they encode directly conditional independence relations at the graph. This makes IDs suitable for the representation of decision problems from the knowledge of human experts, which prefer to assess probabilities in a cause-effect direction.

The main drawback of IDs is that they assume symmetry. IDs are only suitable for the representation of sequential decision problems with a fixed order of decisions and without constraints which condition the appearance of a variable. As explained with detail in Qi et al. (1994), Call & Miller (1990) or Smith et al. (1993) IDs are inflexible and inefficient for the representation of asymmetry. We can confirm this result from the analysis of the representation of the three asymmetric decision problems. IDs represent asymmetry following the approach of an artificial symmetrization, what is very inefficient. The restriction of the outcomes of a chance variable is represented by adding artificial states to the state space and taking degenerated probability distributions to describe that the real states are impossible for certain conditioning scenarios. The restriction of legitimate decision alternatives by information states is achieved by assigning large negative values to the utility states associated to the decision node so that the unavailable decision alternatives are not chosen by the solution algorithm. The approach to use utility nodes to model restrictions on decisions obscures the representation of the structure of the problem as either artificial utility nodes appear which model the restrictions on the decision or the utility node associated to the decision have more variables in its domain. Both the addition of dummy states to the state space of the variables and the modeling of the information states on the utility nodes obscures the structure of the problem and increases the time and space required for the solution as we have shown at the representation of all three problems. Another shortcoming of this approach is that the decision analyst gets no information about conditioned scenarios and constraints from the graphical model.

IDs are also unsuitable for the representation of order asymmetry. As IDs require a total linear order of decisions, it is necessary to include all admissible decision/observation sequences directly in the representation model when the order of decision is undefined. From the representation of the diabetes problem we have seen that order asymmetry is represented introducing variables, which model artificially the order of decisions and how all possible test outcomes must be included at the state space of the test result variable. We have concluded that for this reason the representation of the n-test problem is infeasible with IDs.

2.3.5 Extended ID representation

Extended IDs (extIDs) were proposed by Smith et al. (1993) and belong to the group of the first alternatives to standard IDs. They use a hybrid ID and DT representation. extIDs combine the representation of the uncertainty model of IDs with the description of asymmetric constraints by means of distribution trees of the conditioning scenarios of a variable. This representation for the distribution enables to model asymmetry as they can make explicit the set of conditionally possible and impossible outcomes or alternatives and make explicit the relationship between conditioning information and the conditional distributions assigned to each state of information.

Definition

extIDs are specified at the relational level and functional level. At the relational level they use an influence diagram as a graph to describe probabilistic relationships. At the functional level they use a distribution tree for showing the conditioning scenarios which lead to different atomic distributions. extIDs use a conditioning function to describe the probability distribution assigned in each conditioning scenario. A conditioning function $C_{X|A,B}(a,b)$ maps the set of all possible conditioning scenarios (outcomes from A and B) to a set of atomic distributions.

extIDs can describe structural asymmetry accurately as they can have *coalesced*, *clipped*, *collapsed* or *unspecified distributions*. *Coalescence* happens when the same atomic distribution is shared between different conditioning scenarios. *Clipping* happens when the conditioning scenarios are impossible and in consequence no outcome is possible. A *collapsed distribution* describes conditional independence between variables. A distribution is collapsed when for some subset of conditioning scenarios the corresponding conditional distributions are assigned independently of the outcome of the rest of the conditioning variables. extIDs can also describe some scenarios as unspecified if they are unnecessary for the problem.

Evaluation

The solution method is the computation of the maximum expected utilities by arc reversals and node elimination. The advantage of extIDs is the use of conditioned probability distributions, which simplify the computations of the solution. Features such as clipped, coalesced or collapsed scenarios allow to detect the presence of unnecessary information and can therefore optimize the computations.

ExtID representation of the diabetes problem

At the relational level the representation of the extID formalism is a conventional ID. Figure 2.7 in Section 2.3.4 shows the ID representation of the diabetes problem. At the relational level no information appears about the structural constraints of the problem. The constraints about the outcomes of the test result are described at the functional level with distribution trees, which show the special conditioning scenarios for the *Test Result 1* and *Test Result 2* node.

This section describes the conditional distribution of the chance variable *Test Result 1*, which depends on the decision *Test 1* and the observation *Diabetes*. The distribution tree describes the probability distribution at the part of the atomic distributions and the factors of the probability function at the part of the conditioning scenarios.

Figure 2.15 shows the distribution tree for $P_{TestResult1|Test1,Diabetes}(test1, diabetes)$. The outcomes of this variable are restricted by the decision to perform a test and the type of test, what leads to five distinct atomic distributions. The atomic distributions show only the possible outcomes and describe clearly which states of the test result are available for a test decision.

This distribution tree contains an irrelevant scenario as the test result is independent from the variable *Diabetes* when the decision is not to perform the test. extIDs describe this as collapsed distribution, where the *Test result 1* node is collapsed over the node *Diabetes* given *Test1=no test*.

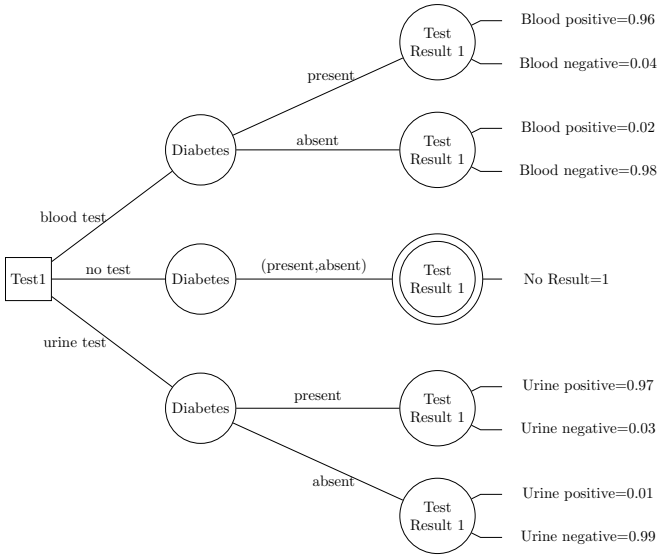


Figure 2.15: Distribution tree for the variable *Test Result 1.*

This section describes the conditional distributions of the chance variable *Test Result 2*, which depends on the decision *Test 1* and the observations *Diabetes* and *Test Result 1*. The conditioning distribution of *Test Result 2* is more complex than the first test result, as an additional constraint with the *Test Result 1* exists. If the second test repeats the first test, the outcome of this test is the same.

Figure 2.16 shows the distribution tree for $P_{TestResult2|Test1,TestResult1,Diabetes}(test1, testResult1, diabetes)$. This distribution tree shows the conditioning scenarios which lead to nine atomic distributions, where only the possible states are shown. The distribution contains some coalesced and collapsed distributions:

- The distribution tree of Test Result 2 is shared between some conditioning scenarios. In the case the second test is different from the first, three sub trees are shared for the conditioning scenario of *Diabetes*.

- In the case the decision is not to perform the test, the test result outcomes (*Test Result 2*) are independent from the variable *Diabetes*, which is described as collapsed distribution.

- In the case the decision is to repeat the test, the outcome of *Test Result 2* is deterministic, because a repetition of a test leads to the same test result as in the previous test. These distributions are collapsed over the variable *Diabetes* as the value of *Diabetes* is irrelevant.

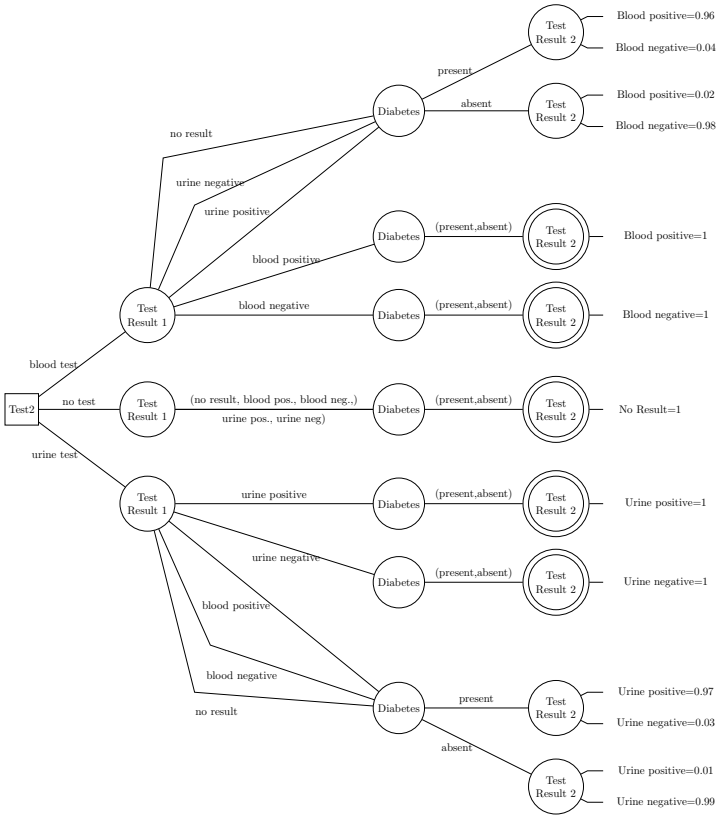


Figure 2.16: Distribution tree for the variable *Test Result 2.*

The extIDs representation improves the ID representation by describing the possible outcomes of the variables with distribution trees. IDs need a potential with 30 values to describe the possible outcomes of *Result of test 1* while extIDs represent this with five atomic distributions. IDs describe the possible outcomes of *Test Result 2* with a potential of 150 values while extIDs use a distribution tree with nine atomic distributions. As extIDs are based on IDs, they also require an absolute order of observations and decisions. Therefore extIDs are not suitable for representing order asymmetry as they need to do handle the artificial variables the ID introduced to model all possible sequences of observations and variables at the model. As we have seen the distribution trees are able to recognize collapsed and coalesced distributions but the underlying problem of order asymmetry remains unresolved. In the case of the n-test problem each test result node would include all test result outcomes from all available tests, which makes the representation complex and untreatable for a larger number of test.

ExtID representation of the dating problem

At the relational level this formalism uses a conventional ID, which is identical to that of the ID representation of the dating problem (see Figure 2.10). At the functional level extIDs use distribution trees to describe the conditional distribution structure of the nodes. The distribution trees are useful as they are able to represent coalesced, clipped and collapsed distributions, which allow for more efficient computations in the solution process.

This section describes the conditional distribution of the decision *NClub* which depends on

the decision *Ask?* and the observations *Accept* and *TV*. The distribution tree describes the set of alternatives for the decision at the part of the atomic distributions and the alternative information states at the part of the conditioning scenarios.

Figure 2.17 shows the distribution tree for $P_{NClub|Ask,Accept,TV}(ask, accept, tv)$. The tree has two atomic distributions, which describes the scenario where Joe decides to go to the nightclub and another scenario, where he does not make any decision. The distribution tree contains a clipped, collapsed and coalesced scenario:

- ExtIDs represent the existence of impossible conditioning scenarios by clipped distributions. A clipped distribution omits the branches for impossible conditioning scenarios, so for example the branches for the options *yes* and *no* are omitted for the node *Accept* , which corresponds to the partial distribution $P(Accept|Ask = no, TV)$.
- The existence of irrelevant scenarios are shown with collapsed distributions. The available decision options of *NClub* are independent from the states of *TV*. In this case the conditional distribution of *NClub* can be collapsed over *TV* given *Ask=no* or given *Ask=yes*. At the distribution tree the collapsed distribution is depicted by showing next to the collapsed node all possible states.
- This tree contains a coalesced distribution because the atomic distribution of *NClub* is shared between two distinct scenarios.

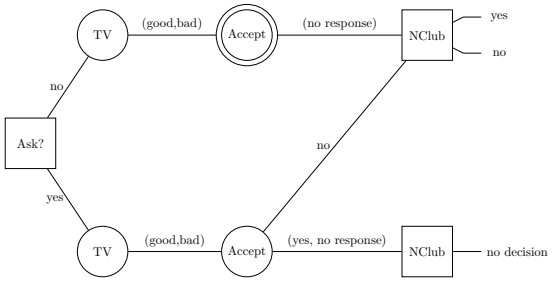


Figure 2.17: Distribution tree for the decision *NClub*.

This section describes the conditional distribution of the utility *TVExp* which depends on the decision *NClub?* and the observation *TV*. The distribution tree describes the expected utilities at the part of the atomic distributions and the factors of the joint utility function at the part of the conditioning scenarios. Figure 2.18 shows the distribution tree for $P_{TVExp|NClub,TV}(nclub, tv)$, which has three atomic distributions with different utilities. The atomic distribution with utility zero is shared between two conditioning scenarios, which correspond to the case Joe decides to go to the nightclub.

This section describes the conditional distribution of the probability distribution of *ToDo* which depends on the decision *Ask?* and the observations *Accept* and *LikesMe*. The distribution tree describes the probabilities at the part of the atomic distributions and the factor of the joint probability function at the part of the conditioning scenarios.

Figure 2.19 shows the distribution tree for $P_{Todo|Ask,Accept,LikesMe}(ask, accept, likesMe)$ which has three atomic distributions with different probability distributions. The distribution tree contains also a clipped, collapsed and coalesced distribution:

- The node *Accept* at the lower branch only shows the option *no response* as the options *yes* and *no* are impossible in this conditioning scenario.

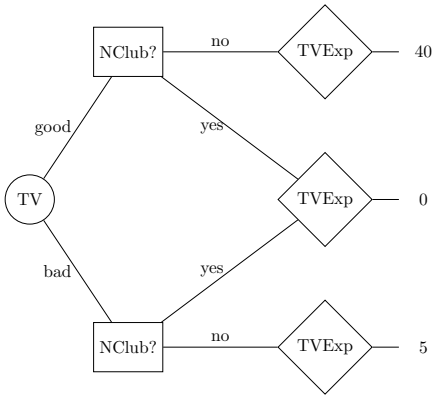


Figure 2.18: Distribution tree for the utility node *TVExp*.

- The node *Accept* at the lower branch is a collapsed distribution as its values are independent of the states of *LikesMe* when Joe decides not to ask for the date (*Ask=no*). In this case the conditional distribution of *Accept* is collapsed over *LikesMe* given *Ask=no*. At the distribution tree the collapsed distribution is depicted by showing next to the collapsed node all possible states.
- The node *ToDo* at the lower branch is a deterministic distribution as the outcome *no decision* happens with certainty. This atomic distribution is shared between three different conditioning scenarios.

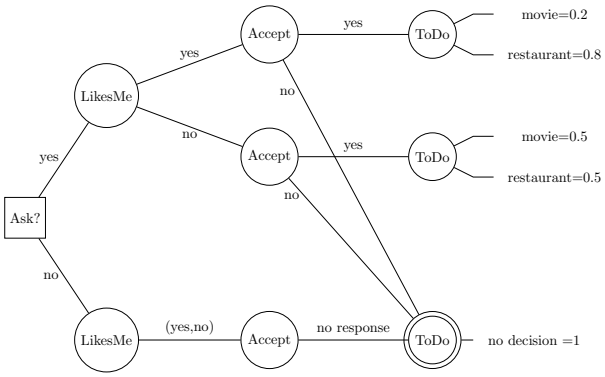


Figure 2.19: Distribution tree for chance variable *ToDo*.

In the previous Section we have explained a conditioned distribution tree for a decision, chance and utility variable in order to describe the semantic of the distribution tree. The dating problem contains more cases of structural constraints than these three examples, but the approach that extIDs follow is the same as explained before. If a variable is non-existent in a decision scenario, the conditioning distribution tree hides the correspondent states for that variable (clipping). In consequence the distribution tree is more compact and only depicts the possible outcomes at the atomic distributions. For example the ID representation needed a potential with 36 values to describe the possible outcomes of *ToDo*. extIDs can describe this in a more compact way as only the possible states are shown at the atomic distributions and collapsing, clipping and coalescence

is used to reduce the structure of the conditioning scenarios (see Figure 2.19). extIDs do not describe the existence of constraints at the graphical model. Therefore from the extID model the decision analyst does not get any information about the existence of the conditioned scenarios of the dating problem.

ExtID representation of the reactor problem

At the relational level the reactor problem is represented as ID (Figure 2.20) and at the functional level it is described with distribution trees (Figure 2.21). The same extID representation can be found in Bielza et al. (2011) and Nielsen (2001) .

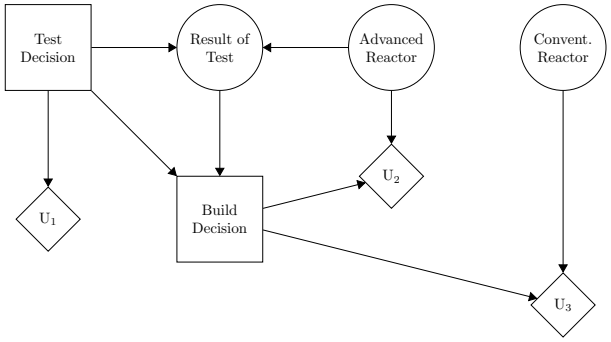


Figure 2.20: ID representation of the reactor problem.

Figure 2.21 shows the distribution trees, which describe the conditional distribution structure of the nodes *Build Decision*, *Result of Test*, U_3(*Benefit of the conventional reactor*) and U_2(*Benefit of advanced reactor*). Distribution trees depict the relationship between conditioning information and the conditional distributions assigned to each state of information and therefore make explicit the possible outcomes and alternatives. The distribution trees of the representation of the reactor problem (Figure 2.21) show the constraint of the problem by using coalescence, clipping and collapsed distributions:

- The available decision options of *Build Decision* depend on the existence of the test result and its value if available. The decision to build an advanced reactor (*ba*) is not allowed when the test result was bad. The distribution tree of *Build Decision* shows the different decision alternatives at the atomic distributions. This tree shows clearly that the decision alternatives are restricted to the set {*bn,bc*} (*build none*, *build conventional*) when the test result was bad. At the contrary all decision options are available when the test result was good or excellent or the test was not performed. The distribution tree describes that three different conditioning scenarios lead to a common outcome by sharing an atomic distribution between three conditioning scenarios. This is a case of a coalesced distribution.

- When the test is not performed, the test results are not available. In this case the conditioning scenarios corresponding to the test results {*b,g,e*} (*bad*, *good*, *excellent*) are impossible. The distribution tree omits the branches of impossible conditioning scenarios, what is called clipping. In the case of the test result, all conditioning scenarios containing *Result of Test* are clipped for the case the test is not performed.

- The existence of irrelevant conditioning scenarios is shown by means of collapsed distributions. The distribution tree for the chance variable *Result of Test* describes the conditioning

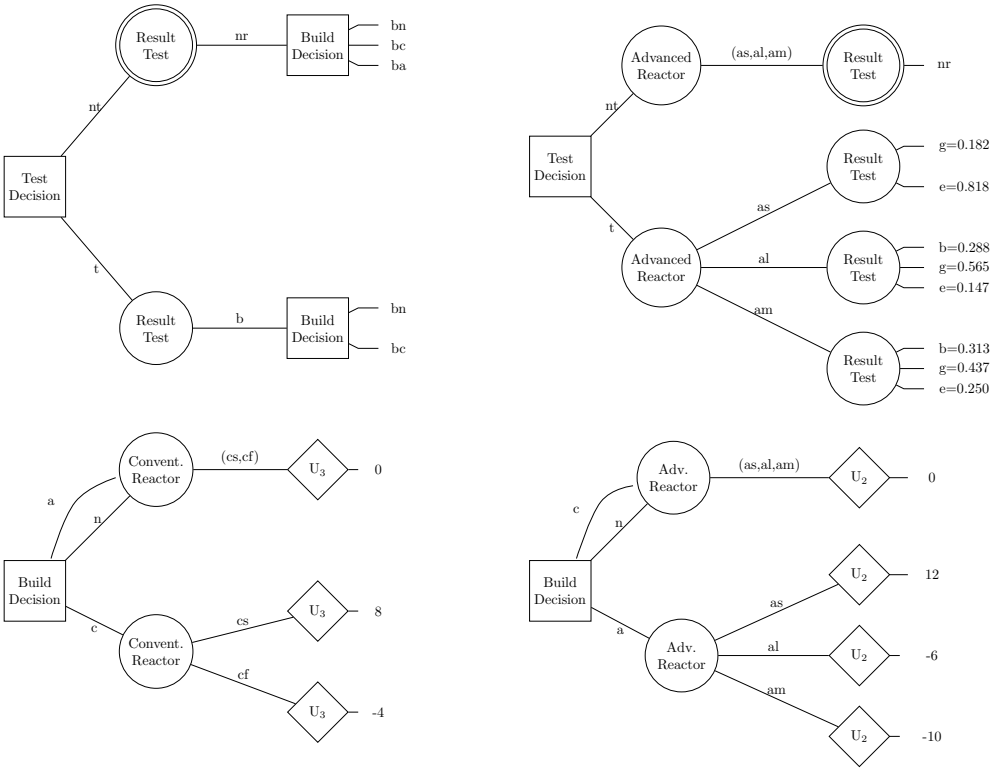


Figure 2.21: Distribution trees of the functional level.

influence of the advanced reactor's components on the test result. In the case the test is not performed the test result is unobserved and is independent of the states of *Advanced reactor*. The distribution tree describes this independence by collapsing the conditional distribution of *Result of Test* across *Advanced reactor* given *Build Decision=nt*.

In the above explanation we have seen that extIDs represent the structural constraints of the problem at the functional level using conditioning functions but not at the relational level. We have seen that extIDs use collapsed distributions to describe irrelevant conditioning scenarios and coalesced distributions to describe information sharing between conditioning scenarios. Both characteristics are used at the computation of the solution to optimize the computations.

Conclusion

ExtIDs are an extended ID representation having the additional ability of describing conditioned probability scenarios. For this reason extIDs have a lot of features in common with IDs: for example they inherit the need to represent order asymmetry with artificial variables which model the sequence of decisions. Although extIDs improve the representation of order asymmetry in certain degree, as we showed in the representation of the diabetes problem, they do not solve it efficiently as all test results of all available tests compose the state space of each test result variable. Therefore extIDs are not suitable for representing order asymmetry as they need to do introduce artificial variables to model all possible sequences of observations and variables.

The representation of the other problems has shown that extIDs are able to describe structural asymmetry. extIDs use coalescence to describe information sharing, clipping to describe

impossible conditioning scenarios and collapsing to describe irrelevant conditioning scenarios. The solution algorithm of the extIDs algorithm is able to exploit these characteristics and to optimize computations by avoiding unnecessary ones. Nevertheless for large probability models an extra effort may be required to determine which distributions should be used in a distribution tree. From the representation of the three problems we have seen how distribution trees make the original ID representation more compact.

extIDs have a strong weakness regarding the description of asymmetry as they do not show the occurrence of asymmetry constraints at the graphical level. Although the asymmetric conditioning scenarios are depicted with full detail at the functional level, the decision analyst does not obtain information about the existence of constraints from the relational, i.e., graphical level.

2.3.6 Sequential valuation network representation

Sequential valuation networks (SVNs) are a formalism for the representation of asymmetry based on valuation networks (VNs). VNs ((Shenoy, 1992)) are an alternative to IDs and DTs as they use arbitrary probability valuations for the description of the probabilistic model, which allow to specify the joint probability with flexibility. While IDs and DTs require the specification of the joint probability distribution in conditional probabilities, VNs use probability valuations which represent factors of the joint probability function and are not necessarily conditioned probabilities. VNs are useful as they relax the requirements for representation and avoid the preprocessing of probabilities. As VNs did not address the representation of asymmetry, Shenoy (1996) proposed a new formalism called *asymmetric valuation networks* (AVNs), which extends VNs with *indicator valuations* for the description of asymmetric constraints. Some time later, Demirer & Shenoy (2006) proposed *sequential valuation networks* (SVNs), which are a combination of features from sequential decision diagrams (SDDs) (Covaliu & Oliver, 1995) and AVNs. This formalism arose as an improvement of some shortcomings of these two formalisms, as described in Bielza & Shenoy (1999), which are mainly the inconsistency of the state space of variables of SDDs, the preprocessing of probabilities required to bring a SDD in a consistent form and the inability of AVNs to express the asymmetric structure of the problem at the graphical level. The resulting formalism, SVN, takes advantage of the relaxed requirements of the description of the probability model of VNs and the expressiveness of the description of the sequential and asymmetric aspects of SDDs.

Definition

SVNs use the description of the probability model of VNs with *valuation nodes* in combination with the graphical features from SDDs to represent asymmetry in a compact way. The SVN contains a sub graph built upon the set of decision, chance and terminal nodes, which is a directed graph with one source node and one sink node. Each directed path from the source node to the terminal node is a possible scenario. The resulting graph is similar to a clustered decision tree, which describes all possible scenarios and indicates constraints by the use of labels above the directed edges.

A SVN representation consists of three part: the graphical part which describes the network structure, the qualitative part, where indicator valuations are specified and the quantitative part, where the numerical details of utility and probability valuations are specified.

At the graphical level SVNs use six different types of nodes, which can be classified as variables and valuation nodes. Variables nodes can be chance or decision (with the same representation as in IDs, with rectangles and circles) or terminal nodes. The arcs between these nodes describe the sequencing of variables in a scenario. Valuation nodes can be classified as indicator, probability

or utility. *Indicator valuations* represent qualitative constraints on the joint space of decision and chance variables and are depicted by double-triangular nodes. The set of variables directly connected to an indicator valuation by undirected edges constitutes the domain of the indicator valuation.

Utility valuations represent factors of the joint utility function (additive or multiplicative) and are depicted by diamond-shape nodes. The set of variables directly connected to a utility valuation constitutes the domain of the utility valuation. *Probability valuations* represent multiplicative factors of the family of joint probability distributions for chance variables in the problem and are depicted by triangular nodes. The set of all variables directly connected to a probability valuation constitutes the domain of the probability valuation. If a probability valuation is conditional then this is indicated by drawing the edges between the probability valuation node and the variables directed towards the variable.

At the qualitative level the indicator valuations are specified in detail. For an indicator valuation ω_1 with domain R, D this specification consists of listing all the allowed states in Ω_1. The use of indicator valuations improves the computational efficiency of the solution techniques as they define the *effective frame* for a subset of variables, i.e., they describe explicitly the compatibility of the values of a variable in certain scenarios. The increased computational efficiency of the solution technique is partly the result of working on effective frames instead of working on general frames. At the numerical level the details of the utility and probability valuations are specified.

Evaluation

The solution method of a SVN is a recursive decomposition of the problem into smaller subproblems. The sub-problems are solved by using a special case of the fusion algorithm and these solutions are recursively combined to generate the solution of the original problem. Inference with VN-based systems is done using two operations called combination and marginalization (Shenoy, 1992).

SVN representation of the diabetes problem

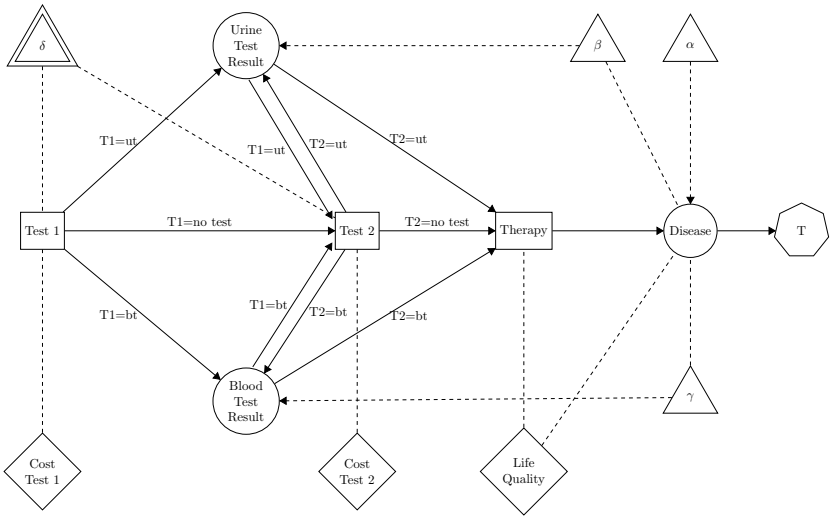


Figure 2.22: SVN representation of the diabetes problem.

Figure 2.22 shows the SVN representation of the diabetes problem as described by Demirer & Shenoy (2006), having the constraint that the same test can not be repeated. In Bielza et al. (2011) appears a similar representation of the problem, but having this representation the possibility to repeat the test.

The SVN diagram of Figure 2.22 represents the sequence of tests explicitly by adding two nodes (*Test 1* and *Test 2*), which represent the choice of the test. The SVN shows the possible scenarios of the test sequences with a directed graph, where *Test 1* is the source node and T is the sink node. The labels above the edges indicate constraints under which a directed arc is possible. The urine and blood test results are only available when such a test is ordered and the test can not be repeated. If the decision is to not perform the test both test results do not appear in the decision scenarios. SVNs model the constraints of the diabetes problem with labels and therefore avoid dummy states.

The constraint of not repeating the test is described with the indicator valuation δ, which specifies the admissible states $(bt, nt), (bt, ut), (ut, nt), (ut, bt)$. The diagram shaped nodes are utility valuations, which represent a factorization of the cost and benefit of diagnosing and treating the patient for Diabetes. The triangular shaped nodes are the probability valuations $\alpha = P(D)$, $\beta = P(Blood\,Test\,Result \mid D)$ and $\gamma = P(Urine\,Test\,Result \mid D)$, which are factorizations of the joint probability distribution $P(D, Urine\,Test\,Result, Blood\,Test\,Result)$.

SVNs use probability valuations, which are multiplicative factors of the joint probability of the chance variables to describe the probabilistic model. Probability valuations are not necessarily conditional probabilities, what gives SVNs flexibility for the description of the probabilistic model. But this may be less intuitive for humans, which are used to appraise probabilities in the cause-effect direction. Further the models for diagnosis problems are built usually on direct probabilities such as prevalence of the disease and the sensitivity or specificity of the test. For this reason the alternative approach of the description of the probability model is not apparent at the SVN representation of the diabetes diagnosis problem, which is built upon direct and conditioned probabilities. Nevertheless, if the model for a diagnosis problem is built from data the flexibility of probability valuations to describe the joint distribution might be a benefit.

The representation of the diabetes problem has shown that SVNs are only able to represent order asymmetry by introducing multiple instances of the test variable and specifying constraints with labels to model the distinct sequences. For a n-test problem the SVN diagram would become very complex as each test result could be observed at each test decision, so that each test result node would have n incoming arcs.

SVN representation of the dating problem

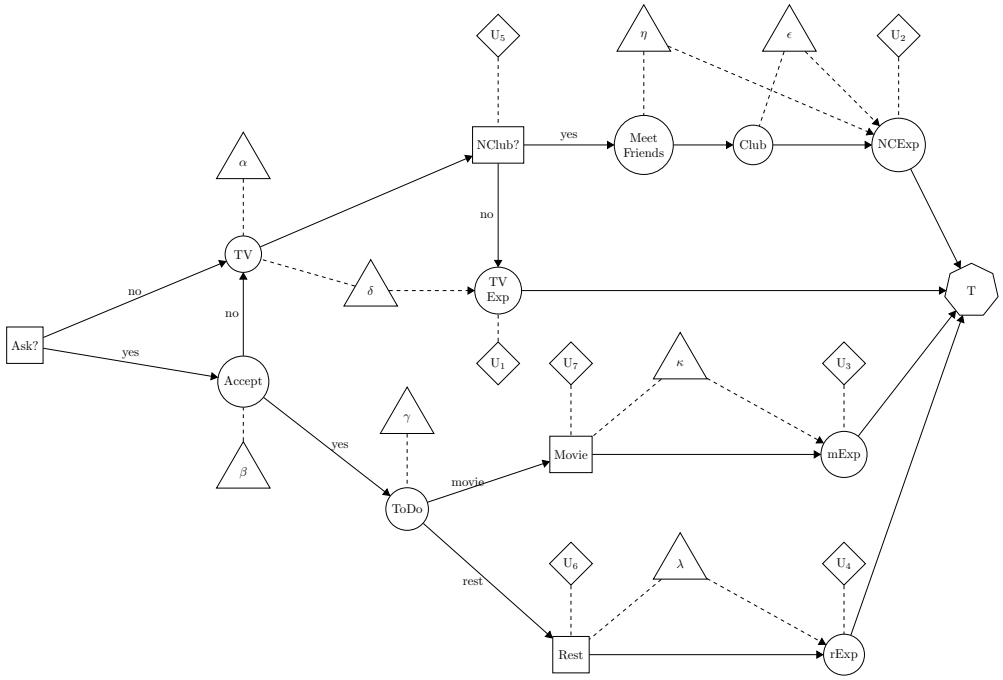


Figure 2.23: SVN representation of the dating problem.

Figure 2.23 shows the SVN representation of the dating problem. The directed sub-graph, which is a compact representation of the decision tree of all possible decision scenarios, describes the sequence of decisions and observations. The labels above the solid line edges describe the structural constraints of the variables, i.e., under which conditions the variable appears. This decision tree based representation implies difficulties to represent unobserved chance variables. These are variables which are not observed in any decision scenario nor appear in the domain of a utility function. In the dating problem the variables *LikesMe*, *mMood* and *rMood* are unobserved. Although these variables have a probabilistic influence on other variables, which are observed at the decision scenario, they do not appear in the SVN representation. This happens as this variables have been marginalized out of the probability distribution as an alternative to representing them at the end of the decision scenario. This requires a pre-processing of the following probabilities: $P(Accept|LikesMe)$ to $P(Accept)$, $P(ToDo|LikesMe)$ to $P(ToDo)$, $P(mExp|mMood)$ to $P(mExp|Movie)$ and $P(rExp|rMood)$ to $P(rExp|Rest)$. Table 2.6 shows the probability distributions described by each probability valuation.

The dating problem contains very different scenarios, which are conditioned on previous decisions or observations. The SVN representation shows that these constraints can be described with labels and no indicator valuations are necessary. This happens as only two variables are involved in the constraints of the dating problem, what can be expressed with labels.

Prob. valuation	Probability distribution	Prob. val.	Probability distribution
α	$P(TV)$	η	$P(NCExp\|MeetFr)$
β	$P(Accept)$	ϵ	$P(NCExp\|Club)$
γ	$P(ToDo)$	κ	$P(mExp\|Movie)$
δ	$P(TVExp\|TV)$	λ	$P(rExp\|Rest)$

Table 2.6: Probability valuations of the SVN representation.

SVN representation of the reactor problem

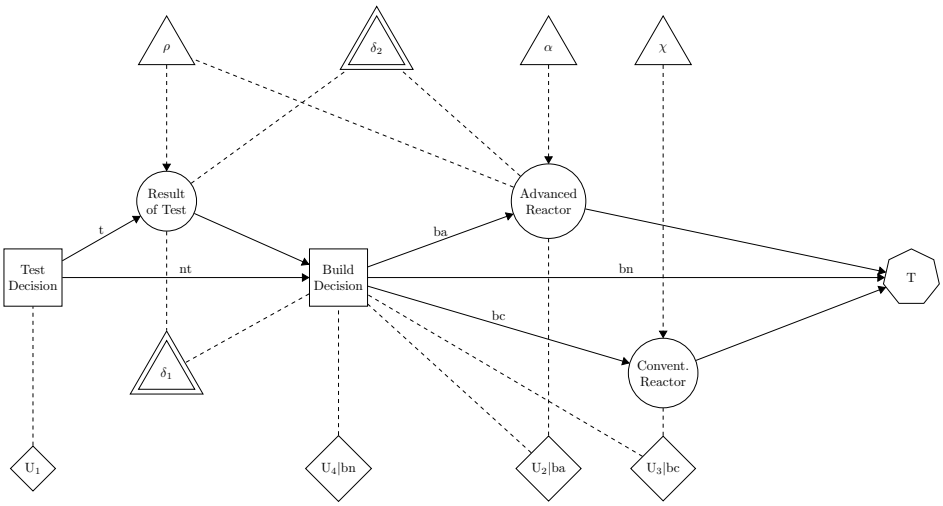


Figure 2.24: SVN representation of the reactor problem.

Figure 2.24 shows the SVN representation for the reactor problem as it appears at Demirer & Shenoy (2006) or Bielza et al. (2011). The SVN diagram uses a directed sub-graph to express the possible scenarios, in conjunction with labels above the solid line arcs, which express constraints. The possible scenarios start from the source node *Test decision* and end in the artificial terminal node T. This graph shows that the test result is only available when the decision is to perform the test by indicating the label t above the arc between *Test Decision* and *Result of Test*. In consequence the state space of *Result of Test* does not contain the dummy state *no result*. In the same way by using labels above the arcs, the SVN diagram describes that the risk of an advanced reactor is only observed when the build decision is to construct an advanced reactor (ba). The same happens for the decision to construct a conventional reactor (bc) or to construct none (bn), the correspondent decision options of the *Build Decision* are shown above the arcs leading into the nodes *Conventional reactor* and T. Labels describe therefore at the graphical level the sequence of observations and decisions explicitly and gives information about constraints, namely under which condition a variable appears at a decision scenario.

The valuation nodes describe the decision problem further by specifying qualitative constraints, utility and probability distributions. These nodes appear at a second sub-diagram with dashed lines. The edges of a valuation define its domain. Thus the probability valuation χ, which describes the risk of a conventional reactor has the domain *Conventional reactor* and the probability valuation α, which describes the risk of an advanced reactor has the domain *Advanced*

reactor. The probability distribution ρ, which describes the test result based on the properties of the components of an advanced reactor is given as its conditioned form $P(\textit{Result of Test} \mid \textit{Adv. Reactor})$. The graph contains several utility valuations, which describe the cost of the test (u_1) and the benefit of the construction for a given type of reactor ($u_{2,}$,u_3, u_4). As there is a constraint between the decision to build a reactor (*Build Decision*) and the event of building the reactor or not, the utility valuation of the benefit of a given type of reactor ($u_{2,}$,u_3, u_4) are given as a valuation fragment conditioned on the correspondent option of the *Build Decision*. For example the benefit of an advanced reactor is expressed as $u_2|ba$ with the domain {*Build Decision*, *Adv. Reactor*}. Fragments allow to specify for which combinations a valuation is allowed. The utility valuation of $\upsilon_2|ba$ describes that the utility values of A are only given for the states *Build Decision* $=ba$, because all other combinations are impossible. SVNs use indicator valuations to express qualitative constraints. The indicator valuation δ_1 with domain {*Result of Test*, *Build Decision*} describes the constraint between the result of the test and the decision options available at *Build Decision*. This indicator valuation numbers out allowed combinations of the *Result of Test* and the *Test decision* and describes that the combination {b,ba}, which corresponds to the decision to build an advanced reactor given a bad test result is not possible. The indicator valuation δ_2 with domain {*Result of Test*, *Adv. Reactor*} describes the constraint between the test result and the properties of the advanced reactor's components. As the test result *bad* statistically does not happen when the components of the reactor indicate success, the indicator valuation of δ_2 omits the state {b,as}.

The SVN representation of the reactor problem has shown that this formalism has the ability to describe structural constraints with labels and indicator valuations. Labels show graphically under which conditions a variable appears in a decision scenario, what is useful if the decision problem leads to very distinct decision scenarios. The SVN representation describes clearly that the observation of the test result only happens when the test was performed and the observations on the risk of an reactor only happens when the decision to build such a reactor was taken. Further indicator valuations can describe qualitative constraints up to any complexity as they can have any number of variables in its domain. The approach for specifying a constraint consists in numbering out all possible combinations, what becomes quite difficult to manage when the domain contains many variables.

Conclusion

In contrast to ID based models, which describe conditional probability models and represent the order of the variables only with a partial order, SVNs combine the flexibility of VNs for the description of arbitrary probability models and the explicit description of the order of variables from SDDs.

SVNs are a hybrid of VNs and SDDs and show at the diagram both a model for the order of decisions and observations and a model for the probability relationships in the same diagram. The tree-like sub graph which presents in a compact form the sequence of decisions and observations for all decision scenarios gives an explicit description of the information flow and is useful for the description of conditioned scenarios. Simple constraints, which comprise only two variables can be described with labels above the edges, as we have seen at the representation of the dating problem. More complex constraints, which involve more than two variables or a partial restrictions, can be described with indicator valuations as we have seen at the representation of the reactor problem. Valuation indicators can get quite complex because they can be associated to several variables, but they have absolute flexibility in specifying any constraint. The approach of indicator valuations to encode constraints is to number out all compatible states, what makes them not easy to understand for a human decision analysts. SVNs use further valuation fragments

to model constraints. Valuation fragments are a compact form of indicating valuations only for those values which satisfy a constraint. The use of fragments requires coherence between labels or indicator valuations and the definition of the fragments. Fragments can be specified on probability valuations and utility valuations.

Although SVNs as explained before do not have troubles to represent structural asymmetry, they have difficulties to model order asymmetry. SVNs require a total ordering of the decisions and observations and therefore must include all admissible decision and observation sequences at the model. Further SVNs use information arcs, what makes it difficult to represent complex problems with SVNs. The combination of the existence of information arcs and the requirement for a total order of the decisions makes SVN unsuitable for the representation of the n-test problem. As we have seen at the representation of the diabetes problem, even this problem with only two tests becomes quite complex.

Another difference of SVNs with respect to ID based formalisms, is the description of the probabilistic model. SVNs rely on valuation networks, which describe the probability model with probability valuations, which are multiplicative factors of the joint probability. Probability valuation are not necessarily conditional or direct probabilities as happens at the ID based formalisms, so the requirements for building a SVN based model are weaker than those of ID based models. While IDs might require a preprocessing of probabilities when probabilities are not given in a direct or conditional form, SVNs can represent a probability model with arbitrary valuations. The possibility of probability valuations to describe the model without conditional probabilities may result convenient when no description of a causal model is available, for instance when the probabilistic model is created from a database. But as humans analyze problems following a cause-effect direction, the representation of non-causal probabilities is not easy to interpret. The SVN representation of the probabilistic model is also modular so that the model is easily adaptable to changes of the decision problem.

As SVNs rely on SDDs they inherit some drawbacks from the decision tree based representation of the decision scenarios. SVNs have difficulties to represent unobserved chance variables. These are variables which are not observed in any decision scenario nor appear in the domain of a utility function. As including unobserved variables in decision trees makes its structure more complex, the alternative is to marginalize these variables out of the probability model, what involves a pre-processing of the probability model. Therefore SVNs usually do not represent unobserved variables, as we have seen at the representation of the dating problem.

We can conclude that SVNs are suitable for the representation of decision problems with structural asymmetry. The approach of SVNs to represent a model for the order of decisions and observations with the model for the probability relationships at the same diagram gives SVN a high explanatory strength. The SVN allows to analyze the explicit sequence of decisions and observations, while the probabilistic relations are also represented explicitly. Before SDDs only the DT formalism gave an explicit description of the decision scenarios, while this formalism was not able to describe probabilistic relations nor present the decision scenarios compactly. Nevertheless the use of the explicit model of the order between variables is also a limitation of the formalism, because for complex problems the construction of an explicit model of order can be quite difficult for a decision analyst.

2.3.7 Asymmetric influence diagram representation

Asymmetric influence diagrams (AIDs) are a formalism presented by Nielsen & Jensen (2000) and is explained further at Nielsen (2001). This formalism is designed for modeling asymmetric decision problems and is based on *partial influence diagrams* (PIDs) proposed by Nielsen & Jensen (1999b). PIDs are a generalization of IDs which allow a non-total ordering of decisions. The idea of relaxing the requirement of IDs for a total linear ordering of decisions to a partial ordering comes from the observation that decision problems have decisions which are conditionally independent and in such a situation it is not necessary to impose a total linear temporal ordering. Nielsen & Jensen (1999b) define a set of conditions which assure that a PID is unambiguous, i.e., it is a well defined representation of a decision problem. PIDs allow a chance node to have several decision nodes as immediate successors and no ordering is imposed on the decisions, what makes them better suitable for the representation of decision problems where the order of decisions is not absolutely sequential. The main contribution of AIDs nevertheless is the adaption of the ID formalism for expressing asymmetric constraints. AIDs give to its components a special semantic for the representation of asymmetry, which is based mainly on a description of the conditions which make an arc or a node appear in a certain decision scenario. AIDs address the representation of structural asymmetry also in greater detail than other formalisms, as they distinguish further *functional asymmetry*. In AID terminology *structural asymmetry* refers to the occurrence of a variable in different scenarios while *functional asymmetry* refers to the restriction of the possible outcomes and decision options of a variable in different scenarios.

Definition

AIDs are based on PIDs and have a specification at two levels: the qualitative level with a labeled graph and the quantitative level with potentials. Structural asymmetry, which describes the occurrence of variables in different scenarios is specified at the qualitative level while functional asymmetry, which refers to the restriction of the possible outcomes and decision options of a variable is specified at the quantitative level. The formal description of an AID comprises the specification at the qualitative and quantitative level:

At the qualitative level a AID is a labeled directed graph $G = (U, E, F)$, where U are the set of nodes, E are the set of arcs and F are the set of labels. The graph of an AID has four types of nodes U: chance, value, *test-decision* and *action-decision* nodes. Chance and value nodes have the same meaning as with IDs. A test-decision (drawn as a triangle) is a decision to look for more evidence, whereas an action-decision (drawn as a rectangle) is a decision to change the state of the world.

The arcs E of an AID have the following meaning. The arcs into value nodes and chance nodes have the same meaning as in IDs. The arcs into decision nodes, termed *informational arcs*, represent a *possible precedence*. In asymmetric decision problems the set of variables observed immediately before taking a decision D may depend on previous decisions and observations, therefore the semantics of an informational arc is redefined to a possible information precedence. An arc between a test-decision node D and a chance node C is termed as test arc. Test arcs mean that the state of D determines whether or not the chance node C is eventually observed. Test arcs can also exist between two decision nodes. In this case a label is used to specify if the arc is an informational arc or a test arc.

AIDs reflect the structural asymmetry graphically by a set of *restrictional arcs* and by a set of labels F. Restrictional arcs are painted with dashed lines. A restrictional arc between the node X and D indicates that the set of legal decision options for D may vary depending on the state of X. The labels, which are associated with a subset of nodes and informational arcs specify conditions under which the associated node or informational arc occurs.

At the quantitative level AIDs contain partial probability potentials to describe the uncertainty associated with a chance variable, partial utility potentials to describe the utility of a value node and restrictive decision functions to describe the legitimate decision alternatives for a decision. AIDs specify functional asymmetry at the quantitative level using these elements. *Partial potentials* for probability or utility specify that a possible outcome of an observation is dependent on the conditioning state. A *restrictive function* specifies that the decision options for a decision variable are dependent on the information state.

Evaluation

AIDs propose a method which allows to identify the parts of the decision problem which are relevant for a particular decision variable. The variables relevant for a decision are variables observed before the decision and also future variables, which may influence the optimal policy for a decision. The rules for determining the relevant variables for a decision are explained in detail by Nielsen & Jensen (1999b). Reducing the original decision problem to smaller sub problems, each one describing the relevant scenario for a certain decision variable, is useful either for the analysis of the decision problem and the solution method. The analysis of the problem becomes easier because a decomposition in smaller sub problems, where only the relevant variables for a decision appear, contributes to a better understanding of the structure of the problem, especially when the problem is complex.

The solution method consists of a decomposition of the original problem into sub problems by considering the existence of distinct informational states described by the observed variables and the value of a restrictive variable (split variable). The decomposition into sub problems yields a collection of smaller symmetric PIDs, which give a complete description of the distinct decision scenarios of the problem. The solution of the collection of smaller sub problems is computational less complex than solving the original problem.

AID representation of the diabetes problem

Figure 2.25 shows the AID representation of the diabetes problem at the qualitative level. The definition of this problem requires a total ordering of the decisions as the test decisions are not independent. According to the definition of the problem, the tests are not repeated and the previous test result is known when taking the next decision. For this reason the AID shows explicitly the order of the tests adding two test-decision nodes, which represent the choice of test and a test result node for each test. The test-arcs between the test decision and its result describe that the result is eventually observed. The labels above the *Blood Test Result* and *Urine Test Result* node express that their states are observed when the test decision is to either perform the blood test (bt) or the urine test (ut). The arcs between the test nodes *Test 1* and *Test 2* and the decision node *Treatment* are informational arcs and describe precedence. This means that the decision regarding the treatment is taken after performing *Test 1* and after performing *Test 2*. The test result of the first test is known when taking the second test decision. For this reason the test result nodes have an information arc leading into the second test decision, which describe a possible information precedence. These arcs have a label to describe a syntactical constraint which breaks the cycle, namely the information precedence between *Blood Test Result* and *Urine Test Result* and *Test 2* does not happen when the observation of the test result corresponds to the second test decision. In the same way the arcs from the second test decision (*Test 2*) to the test result nodes have labels to describe that the same test can not be repeated. The arcs from the test results into the decision node *Treatment* describe a possible precedence, as the test results may not be available.

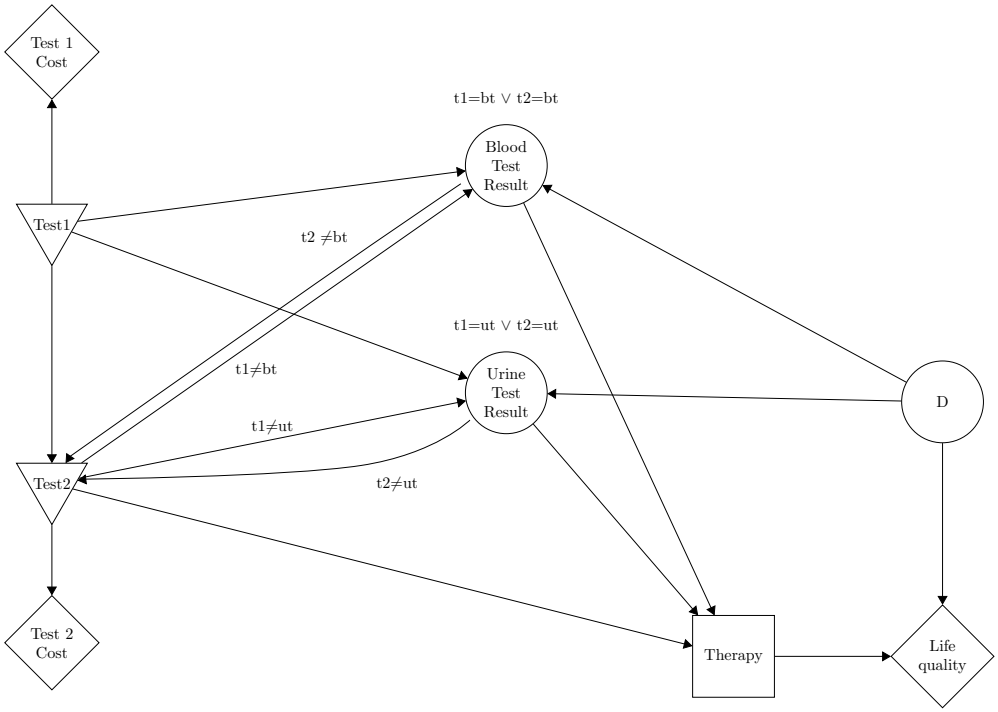


Figure 2.25: AID representation of the diabetes problem.

AIDs have test-arcs which are in principle suitable for the representation of a test problem. Test-arcs are a precise description of the conditioning of the test result on the test decision, namely the evidence of the test result is only available when it is decided to look for more evidence. This works if the test decision only includes one test option. If the test decision considers the choice between several test alternatives AIDs need to use labels to describe to which test decision corresponds a given test result. See the use of labels for the test results of the diabetes problem (Figure 2.25), which is a two test problem. The combination of test-arcs and labels allow AIDs to describe the structural constraint that the test result is not available when the test is not performed and avoid dummy states at the test result variable.

Nevertheless the n-test problem is not easily representable with AIDs as the order of the decisions is represented explicitly at the model and the formalism has informational arcs. Each test decision would need n outgoing arcs to each of the n test results, describing with a label for which test decision the test result is observable. At the same time each test decision (except the first) would have n incoming arcs to describe the possible information precedence of the previous test results. The example of the diabetes problem, which is a two-test problem, makes evident the difficulties of AIDs to represent the n-test problem.

AID representation of the dating problem

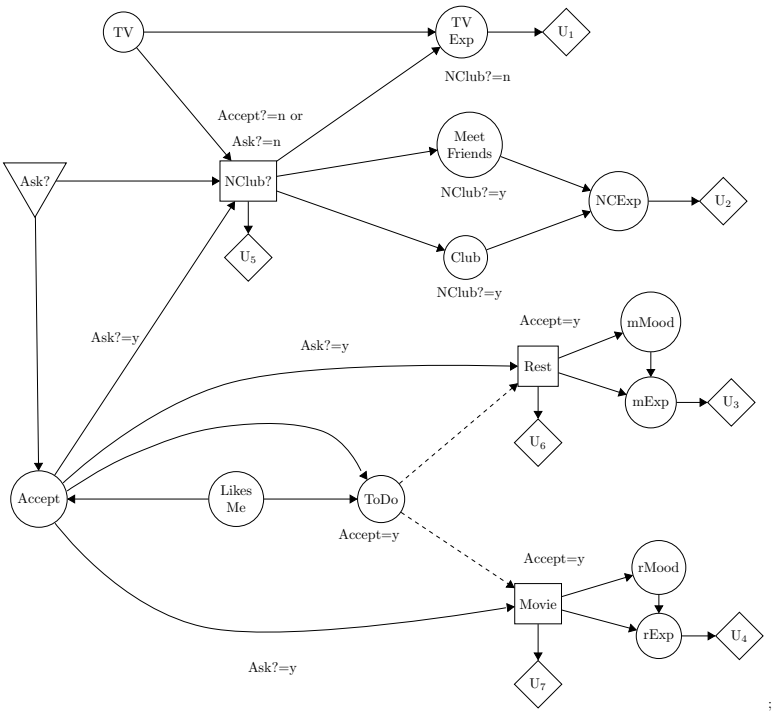


Figure 2.26: AID representation of the dating problem.

Specification at the qualitative level

Figure 2.26 shows the AID representation for the dating problem as represented in Nielsen & Jensen (2000). The diagram shows the representation of the problem with a partial order of the decisions. This leads to a natural representation of the conditioned scenarios as a chance variable can have several decisions as immediate successor and the sequential representation of decisions in a total order is avoided. The adapted semantics of arcs or nodes and the labels at the diagram describe concisely that a variable may be conditioned on a previous event. The combination of the representation with partial order and the use of labels make it possible to describe the asymmetric nature of the dating problem clearly as explained in the following.

The AID diagram shows informational arcs, which describe the possible precedence of information. For example the outgoing arcs from the node *Accept* to the decision nodes *Movie*, *Restaurant* or *NClub* are informational arcs, which describe that the variable *Accept* may be known when taking the decision. The associated label *Ask?=y* specifies that the information arc only occurs when Joe asked for the date.

The diagram shows also restrictional arcs, which are a subset of informational arcs. A restrictional arc appears between the node *ToDo* and *Restaurant* or *Movie* and denotes that the set of legal decision options for the decision may vary depending on the state of *ToDo*. In this sense, if the value of *ToDo* is *restaurant*, no decision options for the decision *Movie* are allowed.

The AID representation shows also test-arcs. The test-arcs between *Ask?* and *Accept* describe that the state of *Ask?* determines whether the state of *Accept* is observed. From the specification of the dating problem we know that the variable *Accept* is only observed if the decision is to ask

for the date ($Ask?{=}y$). In the same manner the test arc between Ask and $NClub?$ describes that $NClub$ is decided upon eventually depending on the state of $Ask?$. The dating problem describes that Joe considers to go to the nightclub if he decides to not ask for the date ($Ask?{=}n$).

Labels at nodes and information arcs describe conditions under which a node or an arc occurs, what in AIDs is termed structural asymmetry. For example the label $Ask?{=}n$ or $Accept?{=}n$ of the node $NClub$ specifies that the node appears only if Joe did not ask for the date or Emily declined the invitation. If a node or arc does not fulfill the conditions of its label it is removed from the decision scenario. This is useful for modeling the decision scenarios conditioned on some previous observations or decisions. For instance Figure 2.27 shows the decision scenario conditioned on $Ask?{=}n$.

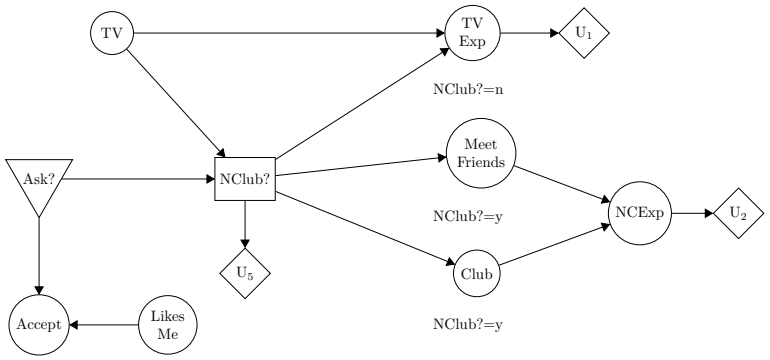


Figure 2.27: Decision scenario conditioned on $Ask?{=}n$.

Labels are also propagated to their successor nodes. For example the nodes *MeetFriends*, *Club* and *TVExp* inherit the conditions specified in the labels of *NClub*. So neither of these nodes appears in the decision scenario of $Accept{=}y$, which is shown in figure 2.28.

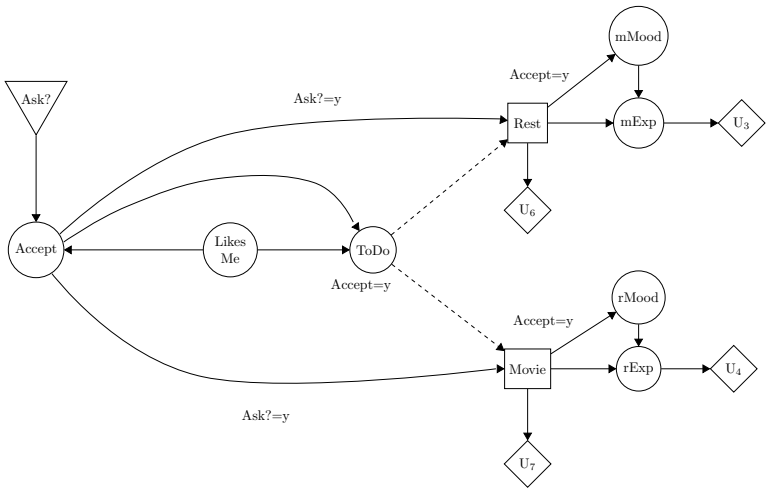


Figure 2.28: Decision scenario conditioned on $Accept{=}y$.

Specification at the quantitative level

At the quantitative level functional asymmetry is specified by restrictive functions and partial potentials for probabilities and utilities. A partial probability potential can specify that given a configuration of the conditioning set for a variable X, some states of a variable X are impossible and therefore undefined (denoted by $\perp$). If a state of a conditioning variable restricts all values of the variable X, the variable X does not appear in a scenario, what is a case of structural asymmetry. This must be reflected at the qualitative level labeling the node X with the states of the conditioning set, which are compatible. For functional asymmetry, which restricts the values of the variable X to a subset without making a variable impossible in a scenario, no labeling at the qualitative level is required.

Further a partial probability potential cannot contain test-decisions in its domain. Regarding utility description a value node X is associated with a partial utility function, where undefined utilities take on the value zero. Further the multiplication and addition of partial probabilities is specified, whereas these operations depend on whether the partial function of the variable is defined or not.

Figure 2.29 shows the partial probability potential associated to the node *TVExp* which has the condition *NClub?=n* associated. This table shows that the values conditioned on *NClub?=yes* are undefined.

TV	good	bad	good	bad
NClub?	yes	yes	no	no
yes	$\perp$	$\perp$	0.2	0.5
no	$\perp$	$\perp$	0.8	0.5

Figure 2.29: Partial probability potential of the variable *TVExp* .

AIDs define also restrictions for decisions at the quantitative level. Each decision variable D is associated with a set of restrictive functions. A restrictive function is associated with a restriction arc between X and D and specifies the legitimate decision options for D given the configuration of the node X. The dating problem contains the restrictive functions for the decision *Movie* and *Restaurant* depicted at figure 2.30, which show that the decision options are restricted by the values of *ToDo*.

γ(ToDo)	Movie
restaurant	{}
movie	{romantic, action}

γ(ToDo)	Restaurant
restaurant	{cheap,expensive}
movie	{}

Figure 2.30: Restrictive functions related to the decisions *Movie* and *Restaurant.*

Description of decision scenarios

AIDs provide a method for decomposing the original problem into sub problems, which describe distinct decision scenarios. This representation is useful either for decision analysis and computations during the solution process. The original problem is decomposed in a collection of smaller sub problems by taking into account the existence of different informational states, which are defined by determining the variables actually observed before taking a particular decision. The different informational states are treated by means of reductions of the original diagram to smaller AIDs. The reduction is based on the value of a split variable, which forms part of the informational state (also termed split configuration) of the new AID. The first step of the

decomposition is to determine the split variable, which is a restrictive variable and may appear at the domain of a label. An AID is assumed to have a unique split variable. The reduction based on the value of the split variable eliminates nodes and arcs according to the evaluation of its labels. This reduction is then applied iteratively to the new AIDs until each AID is reduced to a symmetric PID, which is the description of a decision scenario.

The decomposition of the AID into smaller AIDs creates a decomposition tree, where every node corresponds to a smaller AID. Figure 2.31 shows a part of the decomposition tree of the dating problem starting from the AID with the split configuration Ask?=n, where the distinct decision scenarios for the variable NClub appear as leaves.

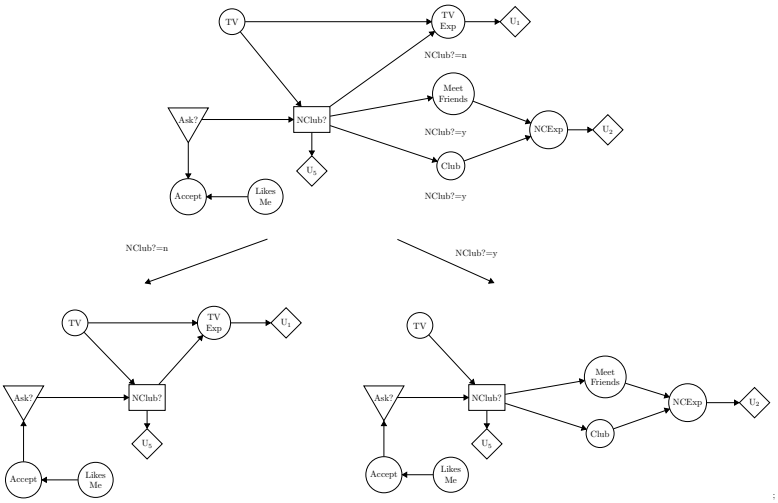


Figure 2.31: Decision scenarios of the decision *NClub*.

AID representation of the reactor problem

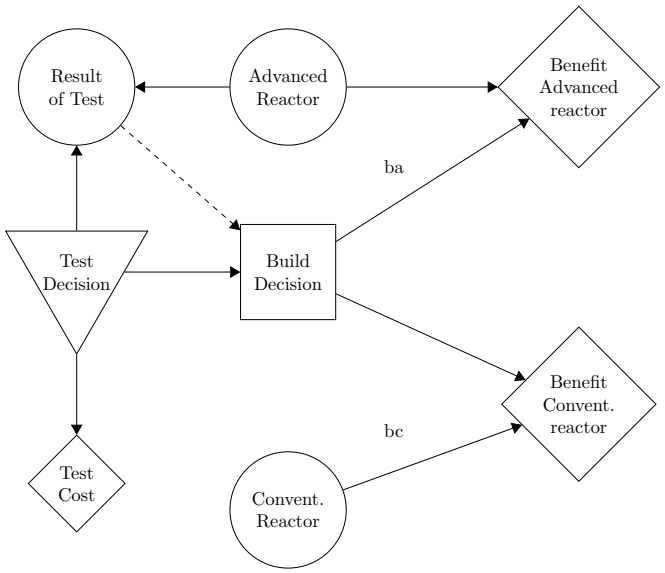


Figure 2.32: AID representation of the reactor problem.

Figure 2.32 shows the AID representation of the qualitative level of the reactor problem according to Nielsen (2001). The decision to evaluate the components of the advanced reactor are represented with the *Test Decision* node, which describes by means of a test-arc that the result is only observed eventually. The arc between the *Test Decision* and *Build Decision* denotes precedence as the decision about performing the test is taken before. The arc between the *Result of Test* and the *Build Decision* is a restrictive arc, which is a special type of information arc. This arc describes first the possible informational precedence of the test result and next the restrictions of some tests results on the build decision. The labels above the arcs from the *Build Decision* to the utilities nodes (*Benefit of advanced reactor* or *Benefit of conventional reactor*) describe for which build decision the utility is observed.

At the quantitative level the restrictions are specified further as explained in the following. Table 2.7 shows the restrictive function of the *Build Decision*, which describes with detail the restrictions of the test result on the decision to build a reactor.

γ (Result of Test)	Build decision
bad	{conventional, none}
good	{advanced, conventional, none}
excellent	{advanced, conventional, none}

Table 2.7: Restrictive function of the *Build decision.*

The benefit of constructing a certain type of reactor are described with partial utility potentials at the quantitative level. These potentials contain zero values for impossible combinations, for instance the utility potential of the *Benefit of advanced reactor* contains zero values for the combinations corresponding to the decision to build a conventional reactor or none. The use of zero values to describe impossible utilities is problematic when the problem contains negative

utilities. In this case the states having a zero utility value are not the worst case and the solution algorithm has no means to discard these states. For this reason AIDs require positive values for the partial utility potential and a transformation of the original utility values to positive values is required.

The restriction between the property of the advanced reactor and the test result, namely that the test result is not bad if the properties indicate success is described with the partial probability potential shown in Table 2.8. AIDs classify this constraint as functional constraint as it describes a restriction on the possible outcomes of a chance variable. AIDs describe this constraint only at the quantitative level and not at the qualitative level.

Advanced reactor	success	limited accident	major accident
bad	$\perp$	0.288	0.313
good	0.182	0.565	0.437
excellent	0.818	0.147	0.250

Table 2.8: Partial probability distribution of *Result of Test*.

Conclusion

AIDs are a special type of ID for the representation of asymmetric decision problems. First AIDs take advantage of the representation of decisions without a total order based on the idea of PIDs to represent conditional independent decisions with a partial order. This gives AIDs the possibility to show the decisions without a complete sequence what is a more natural representation in an asymmetric decision problem, where the occurrence of a decision may depend on previous observations or decisions. AIDs adapt the ID formalism further by giving to its components a special semantic for the representation of asymmetry. First AIDs use an adaption of informational arcs, which describe a possible information precedence and allow therefore the possibility that the variable is not available when taking the decision. Next AIDs use restrictional arcs to describe that the legal decision options for a decision may vary depending on the informational state. Another component for the description of asymmetry are labels associated with nodes or informational arcs, which describe under which conditions the node or the informational arc occur. Test-decisions and test-arcs describe the decision to look for more evidence, what is useful for the description of information gathering patterns or diagnosis problems, where the acquisition of information has a cost and therefore a decision to decide upon assuming this cost must be included in the model. From the representation of the dating problem we have seen how all these components — information arcs, test-decisions, restrictional arcs and labels — are used to describe the conditioned scenarios and avoid the use of dummy states. AIDs are suitable for the representation of structural asymmetry and distinguish further between the fact that the occurrence of a variable is conditioned and the fact that the possible outcomes of a variable or the decision options of a decision are conditioned. The first type of asymmetry is described at the qualitative level with labels and is called structural asymmetry and the second type of asymmetry is called functional asymmetry and is described at the quantitative level with partial potentials (it is not represented always at the qualitative level). At the dating problem appears only the first type of asymmetry but from the representation of the reactor problem we have seen how functional asymmetry is expressed by means of the example of the restriction of the test result on the test decision.

The use of labels at the qualitative level, which describe conditions under which a variable appears (i.e. structural asymmetry) is useful as the decision analyst can read information about conditioned scenarios directly from the graphical model. This information about conditions at

the graphical level can be used further to break the original problem into smaller sub-problems, so that the decision analyst can focus on a specific decision or see the different decision scenarios, what improves the understanding of the decision problem, especially when the problem is complex. But the use of labels can make the graphical model also more difficult to read when the problem gets complex, especially when the nodes have large state spaces.

The representation of order asymmetry is described at the representation of the diabetes problem. AIDs relax the requirement for a total ordering of decisions, which is required at IDs to a partial order, when the decisions are independent. Therefore the representation of order asymmetry depends on the definition of the problem and whether the problem can be represented as well-defined AID. A representation of the diabetes problem, where the test decisions are represented directly at the model would not be a well-defined AID, because the explicit representation of information precedence with information arcs would lead to a (unbroken) cycle in the diagram. Therefore a representation with artificial nodes, which model the order of the decisions is needed. From the representation of the diabetes problem, which is a two test problem it became evident how complex would become the representation of the n-test problem with AIDs. We can conclude that the representation of information precedence with information arcs are the main reason why AIDs are unsuitable for the representation of order asymmetry. The representation of the diabetes problem has shown also some limitations of test-decisions and test-arcs. Test-arcs are in principle suitable to model a test problem, but they are unsatisfactory when the decision is not binary, namely when the decision has several options a label is necessary to specify to which decision the chance variable corresponds.

2.3.8 Unconstrained influence diagram representation

Unconstrained influence diagrams (UIDs) were first presented by Jensen & Vomlelová (2002). UIDs are an extension of IDs designed to describe decision scenarios where the order of decisions and observations is only partially specified. UIDs describe such problems with a partial temporal order at the model and are therefore suitable for the representation of diagnosis or troubleshooting problems.

UIDs were proposed because other formalisms known so far could not represent efficiently order asymmetry. Also the PID framework was not completely satisfactory as it only allows to represent decisions with a partial order when the representation of the problem is a well defined PID (Nielsen & Jensen, 1999b). This means that the partial order of decisions represented in the diagram must yield the same maximum expected utility when extended to a linear order. Because of this limitation UIDs were proposed as they allow to represent decisions with a partial order independent of the maximum expected utility of each of the linear extensions. It is the solution algorithm, which is concerned with finding the optimal decision policy and the best ordering of variables and decisions..

Definition

The specification of UIDs is similar to that of IDs. At the qualitative level a UID is a DAG over three sets of nodes: a set of decision nodes D (represented with rectangles), a set of chance variables C (represented with circles) and a set of utility nodes U (represented with diamond shapes). UIDs describe explicitly when a chance node is *observable* (drawn with a double circle). Non-observable chance nodes do not have decision nodes as children. A chance node becomes observable when all its preceding decisions are taken. The meaning of the arcs is similar to that of IDs. An arc into a decision variable represents informational precedence, an arc into a chance variable represents causal influence and a link into a utility variable represent functional dependence.

The representation of the quantitative level is very similar to that of IDs. There are probability potentials associated to chance nodes and utility functions associated to utility nodes. The local utility functions are additive factors of the total utility function. The UID represents a partial order of the decisions and observations. The extension to different linear orders from the partial order forms part of the solution process.

Evaluation

Although UIDs allow partial order of observations and decisions during the modeling phase, the absolute order has to be considered during the solution phase. The solution of an UID (finding the optimal strategy), is more complex than in the case of IDs. In UIDs the solution should reveal the optimal decision policies but also the best ordering of variables and decisions. A possible solution method of a UID could be to unfold it into a decision tree and compute the optimal strategy. The authors of the formalism propose a more efficient solution based on the construction of a S-DAG. This approach has the following steps:

- Define a partial temporal order from the structural specification, which is a graph representing the possible optimal temporal sequences of observations and decisions. This graph contains a subset of all possible sequences of observations and decisions. This happens as different orderings of cost-free observations or sequences of variables of the same type do not influence the expected utility (EU) and therefore different sequences can be equivalent.

- Construct a strategy-DAG (S-DAG), which is a directed acyclic graph representing possible conditional orderings of the variables and calculate the set of step-policies and decision-policies. The strategy of a S-DAG consists of a step-policy for each node and a decision policy for each decision. A step-policy is a rule for each node that, based on the predecessors of the node, specifies to which of its successors to go. The decision policy is a rule that based on the predecessors of the node specifies which state of the decision to take.

- Given the S-DAG there exist two approaches to compute the optimal strategy: The first is based on variable elimination and the second is to unfold the S-DAG into a strategy tree and compute the expected utility from it.

Some time later two alternative solution methods for solving UIDs were proposed: the any space algorithm of Ahlmann-Ohlsen et al. (2009) and the anytime algorithm of Luque et al. (2010).

UID representation of the diabetes problem

Figure 2.33 shows the UID representation of the diabetes problem. The test options are represented directly and the observable chance nodes are represented explicitly with a double circle. The variable symptom can be observed without any cost. The variables *Blood test result* and *Urine test result* become observable when the preceding decision has been taken. This example shows how the representation with a partial order of the decisions allows to represent the test decisions directly in the model. As consequence of modeling directly the specific test decisions also the test results are specific, i.e., the state space of the *Result of Test* has only the possible outcomes of one test. But as UIDs do not describe structural constraints explicitly, they need to use dummy states to represent the fact that no result is available when the decision is to not perform the test. Therefore the state space of the test result contains an additional dummy state, as shown in Table 2.9.

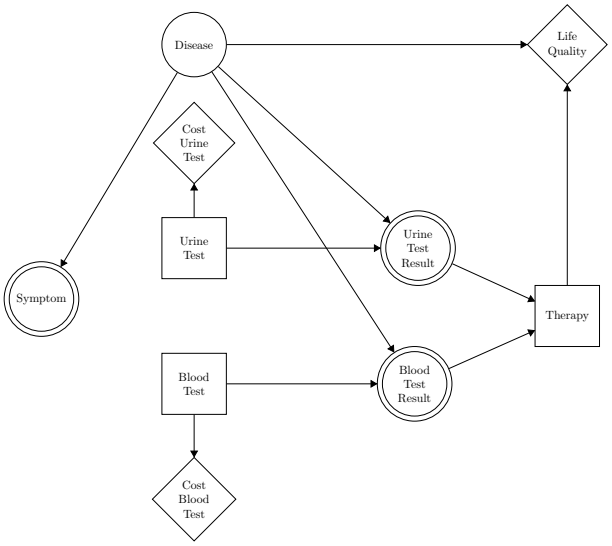


Figure 2.33: UID representation of the diabetes problem.

Blood Test	not test	test	not test	test
Diabetes	absent	absent	present	present
present	0	0.02	0	0.96
absent	0	0.98	0	0.04
no-result	1	0	1	0

Table 2.9: CPT of the variable *Blood Test Result* .

A description for the resolution of this problem with different algorithms can be found at Luque et al. (2010). One of the solution approaches described there consists in calculating the set of step-policies for the nodes *Symptom*, *Blood Rest Result* and *Urine Test Result* and the decision-policies for the decisions *Blood Test*, *Urine Test* and *Therapy* and calculate the expected utility by unfolding the strategy into a strategy-tree. The S-DAG for the diabetes problem, which represents all possible conditional orderings of the variables is shown in figure 2.34. This graph shows that there exist only two admissible orders. Admissible orders are the extension of the partial order of the UID to a linear order, what yields a influence diagram.

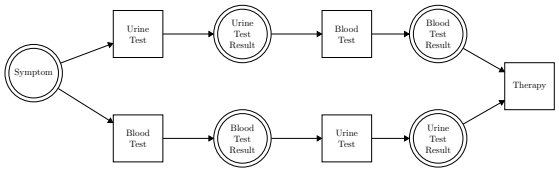


Figure 2.34: S-DAG of the diabetes problem representation

At the literature appear two representations of this problem: Jensen et al. (2006) has a very similar representation and Bielza et al. (2011) shows a slightly different one. The differences of the latter regard the use of auxiliary variables, but not the use of the UID approach for the representation of the problem.

From the representation of the diabetes problem we have seen that UIDs are suitable for the description of diagnosis problems as they represent clearly the relation of the decision to make a test and the information which is revealed from this test. UIDs admit a partial order of the decisions what makes it possible to model order asymmetry efficiently. Further UIDs do not have information arcs as the order of the decisions is partial. For this reason UIDs are suitable to represent the n-test problem as they can represent each test decision directly with a relation to the respective test result.

UID representation of the dating problem

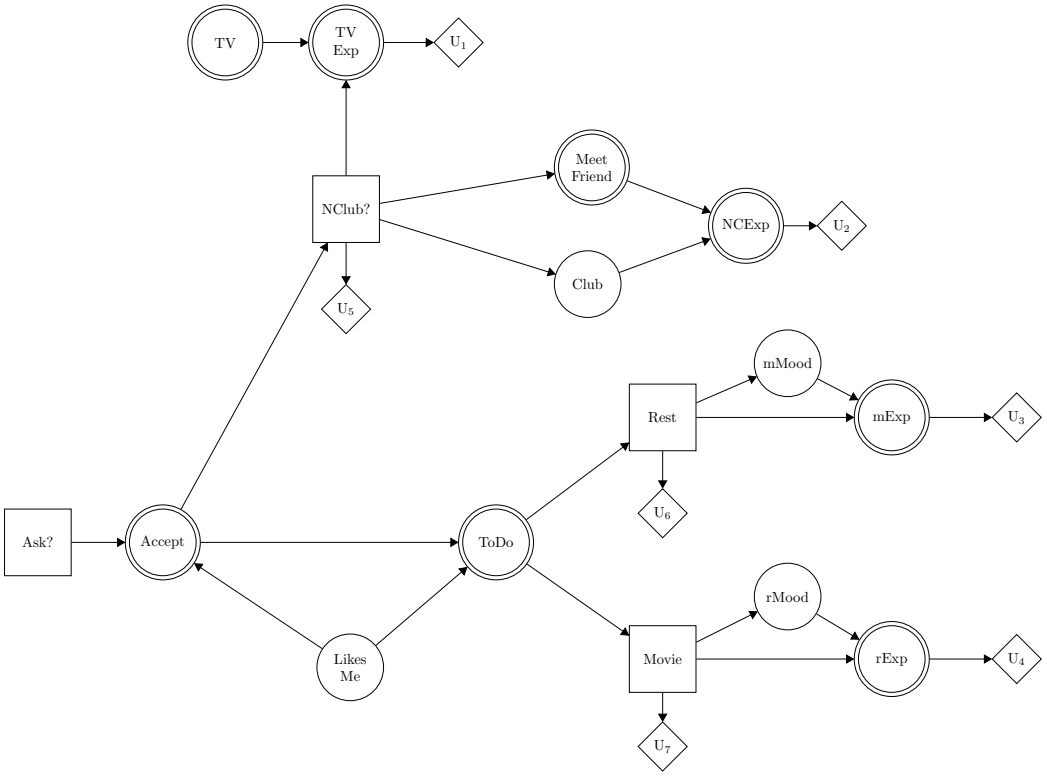


Figure 2.35: UID representation of the dating problem.

Figure 2.35 shows the UID representation of the dating problem. The dating problem contains several structural constraints as the decision scenarios are very different depending on the state of previous observations and decisions. As these constraints are modeled implicitly at the quantitative level with dummy states and degenerated probability and utility functions, no information about structural constraints is available at the graphical model of the UID. The state space of the variables include the same dummy states as those described in Table 2.4 for the ID representation (see Section 2.3.4) .

The UID representation shows explicitly which variables are observable, but this information is not relevant for the description of the dating problem. The variable TV is observable without any action. The variables *Accept* and *ToDo* correspond to Emily's responses and are therefore observable. These variables become observed when the decision about asking for the date was

taken (*Ask?*). The variable *LikesMe* is not observable as it corresponds to a feeling of Emily. In the same sense the variable *Club* may not be observable as it corresponds to Joe's valuation of the quality of the music, etc. .On the contrary the variable *MeetFriends* is observable. The variables *TVExp*, *NCExp*, *mExp* and *rExp* are observed as they determine Joe's satisfaction with spending the event in a certain way.

UID representation of the reactor problem

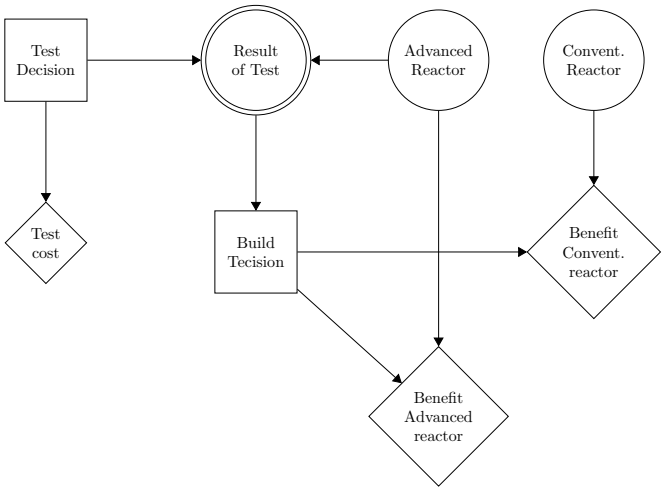


Figure 2.36: UID representation of the reactor problem.

Figure 2.36 shows the UID representation of the reactor problem. The variable *Result of Test* becomes observable when the test decision is taken. The properties of the advanced reactor and conventional reactor are not observable, therefore the correspondent variables are not observable. The reactor problem contains only structural constraints but not order asymmetry. As structural constraints are not explicitly representable with UIDs, the representation of the reactor problem includes dummy states at the state space of the variables and degenerated probabilities and utilities. The specification of the quantitative level is identical to that of the ID representation of this problem explained in Section 2.3.4. In consequence the UID representation has the same troubles as the ID representation because of the use of dummy states.

Conclusion

From the representation of the diabetes problem we have seen that UIDs can represent efficiently order asymmetry as they describe the decisions with a partial order at the model, what allows to represent the test decisions directly. UIDs represent diagnosis problems in a compact form, as they represent clearly the relation of the decision to make a test and the information which is revealed from this test. Further UIDs admit a partial order of decisions and they use *observed variables* as alternative approach to information arcs to describe information precedence. Collectively its ability to represent order asymmetry and the absence of information arcs make UIDs appropriate for the representation of the n-test problem.

UIDs are the only model seen so far that have an alternative approach to information arcs to describe information precedence. Nevertheless this shows also a limitation of UIDs. UIDs assume

that a variable is observed only when all its parent decision have been made. Therefore they cannot represent problems in which a variable is observed as soon as any of its parent decision is made. This is a limitation of UIDs, because there are IDs for which an equivalent UID does not exist. This limitation does not happen for any other formalism discussed in this chapter.

Although UIDs are suitable for the representation of order asymmetry they have troubles to represent structural asymmetry. As UIDs have no means to describe structural constraints explicitly, they follow the inefficient approach of IDs of an artificial symmetrization with dummy states. For this reason UIDs are unsuitable for representing decision problems with structural asymmetry as we have seen at the representation of the dating and reactor problem.

2.3.9 Sequential influence diagram representation

Sequential influence diagrams (SIDs) were first presented by Jensen et al. (2006). This formalism combines features from the asymmetric influence diagrams of Nielsen & Jensen (2000), sequential valuation networks of Demirer & Shenoy (2006) and unconstrained influence diagrams of Jensen & Vomlelová (2002). SIDs are well suited to model structural and order asymmetry. The SID representation improves on the SVNs representation by using influence diagrams to represent uncertainty, allowing unspecified/partial temporal orderings, like in UIDs, and allowing chance nodes that do not appear in any scenario, like in AIDs.

Definition

A SID represents in the same diagram a model for the description of uncertainty and a model for the description of information precedence and asymmetric constraints. This approach is similar to that of SVNs, but the uncertainty model of SIDs, which is not based on VNs, represents conditional probabilities.

The graph of a SID is composed by three type of nodes: chance nodes (drawn as ellipses), utility nodes (drawn as diamonds) and decision nodes (drawn as rectangles). The first model describes with continuous arcs probabilistic dependence relations for chance nodes (also referred to as *conditioning arcs*) and functional relations for utility nodes. The second underlying diagram of a SID encodes informational precedence, structural and order asymmetry. This diagram is a compact sequence diagram, which describes under which conditions the decisions and observations appear. This information is represented with dashed lines, which are called *structural arcs*. Structural arcs encode the structure of the decision problem, such as information precedence and asymmetry, by using annotations. An annotation $g(x, y)$ of a structural arc between the node X and Y (also termed guard) describes the condition under which the next node in the set of scenarios is the node that the arc points to; when the condition is fulfilled the arc is said to be *open*. If there are constraints on the choices at any decision node, then this is specified at the second part of the annotation. The set of scenarios defined by a SID can be identified by iteratively following the open arcs from a source node (node without incoming arcs) until a node is reached with no outcoming arcs. The SID formalism distinguishes between observable and non-observable variables. Observable variables are associated to structural arcs. A variable is said to be observable if there is at least one decision scenario in which the state of X is observed. SIDs use clusters to describe partial temporal orderings. Clusters are a collection of nodes where the temporal ordering of the nodes inside the cluster is not totally specified. The cluster itself is treated as a single node in terms of information precedence in reference to the preceding and subsequent nodes. Clusters allow the temporal ordering of decisions to be unspecified. This approach is the same as that of the UID representation, where the specification of the order of the decisions is postponed to the solution phase and is not considered at the modeling phase.

This implies that the solution of the SID looks not only for an optimal strategy for the decisions but also for an optimal conditional ordering of the decisions.

Evaluation

The solution approach for a SID is a decomposition of the asymmetric problem in smaller symmetric sub-problems, which can be solved recursively. The symmetric sub-problems are organized in a *decomposition graph*, which is constructed by following the temporal ordering of the variables and which is used as computational structure for organizing the sequence of variable eliminations of the solution process.

SID representation of the diabetes problem

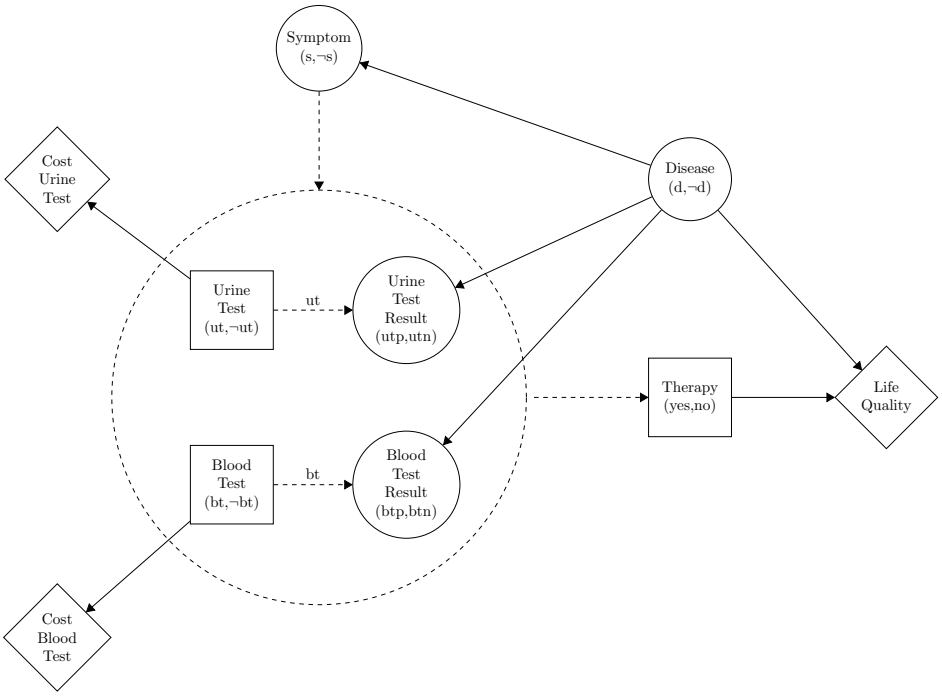


Figure 2.37: SID representation of the diabetes problem.

The diabetes problem is represented as SID in the literature in Bielza et al. (2011) and Jensen et al. (2006). Figure 2.37 shows the SID representation of the diabetes problem according to Jensen et al. (2006). This representation shows a cluster of nodes (depicted with a dashed circle) which describes a part of the model with a partial temporal ordering, namely the partial ordering of the *Blood Test*, *Urine Test* and the related test result variables. This cluster describes that the test decisions precede the respective test results (*Blood Test*$<$ *Blood Test Result* and *Urine Test* $<$ *Urine Test Result*), but that the ordering of the decisions is unspecified. The outgoing structural arc from the cluster to the decision node *Therapy* means that the test decisions happen before the decision of the therapy is made.

The structural arcs between a test decision and the test result node mean that the result is only revealed if the test is performed. This condition is described with a label above the

structural arc. In the diabetes problem the specification of the condition is very simple, because the test node is the first node in the model, but the description for the condition formally includes the sequence of previous decisions and observations which make the arc relevant.

SIDs follow the approach of UIDs of admitting partial temporal orders for parts of the problem, where the order of decisions is unspecified. This solution avoids to describe explicitly in the model the possible decision sequences. This makes SIDs suitable for the representation of order asymmetry and diagnosis problems. Regarding the representation of the n-test we can say that SIDs can easily represent this problem, as they follow the same approach as UIDs, which are suitable for the n-test problem. The representation of the n-test problem would be an extension of the model shown at figure 2.37, where the n-tests and their respective results would be represented inside the cluster, with unspecified temporal order.

SID representation of the dating problem

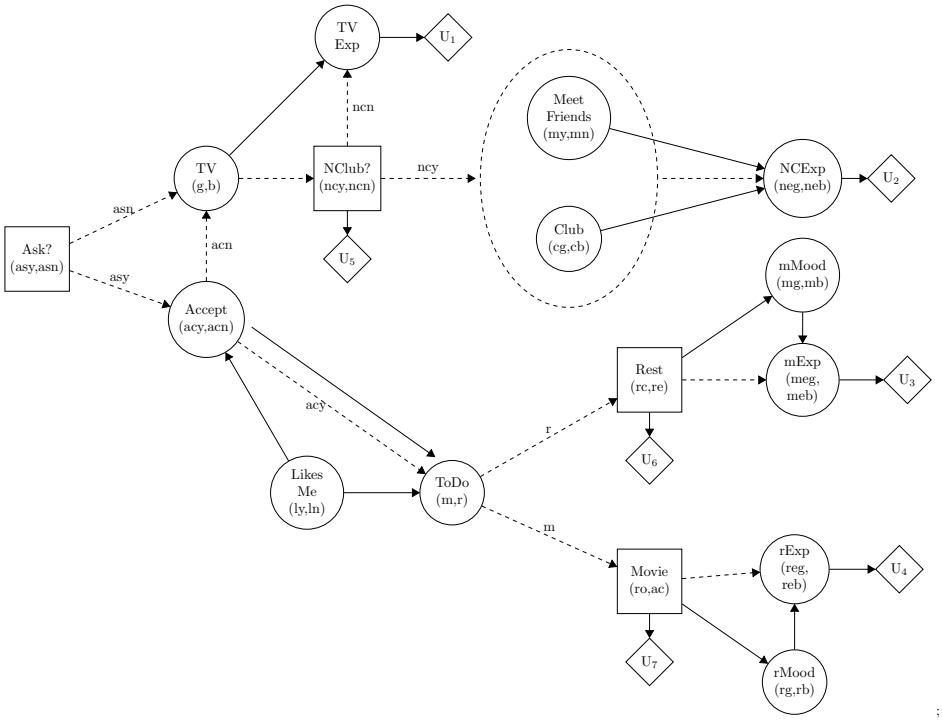


Figure 2.38: SID representation of the dating problem.

Figure 2.38 shows the SID representation of the dating problem according to Jensen et al. (2006). The structural arcs describe explicitly the very different scenarios of the dating problem. Starting from the *Ask?* node, which does not have any incoming arc, the scenarios can be built by following the open arcs. The structural asymmetry becomes very clear as the set of scenarios is very different depending on the value of decisions and observations, which are annotated above the arcs. For example, the set of scenarios for the decision option *Accept=no* is very different from that of the decision option *Accept=*yes. Another example of the existence of disjoint scenarios are the annotations of the arcs outgoing from the node *ToDo*, which describe a disjunction.

The graphic model also contains conditioning arcs, which denote causal influence. The node

Accept has a casual influence on *ToDo* and is an informational predecessor, what is described with an conditioning arc and a structural arc. The representation of the dating problem contains also a cluster of nodes (depicted with a dashed circle) which describe parts of the problem with unspecified temporal order. The order of the nodes *MeetFr* and *Club* are unspecified, but their overall position in terms of information precedence is specified as they are observed after *NClub* but before *NCExp*. The use of both structural arcs and conditioning arcs allow the SID to describe the problem without merging their semantics. The cluster formed by *Club* and *MeetFr* uses both conditioning arcs to describe the probabilistic influence on *NCExp* and structural arcs to describe the informational precedence of the cluster in reference to *NCExp*. The representation of the dating problem also shows variables which are neither observed at any decision scenario and are not part of the domain of any utility function. So the variables *LikesMe*, *mMood* and *rMood* appear only as they play a role for mediating the probability of *mExp* and *rExp* respectively. The representation of these unobserved variable at SIDs are an improvement with respect to SVNs, as explained at Section 2.3.6.

SID representation of the reactor problem

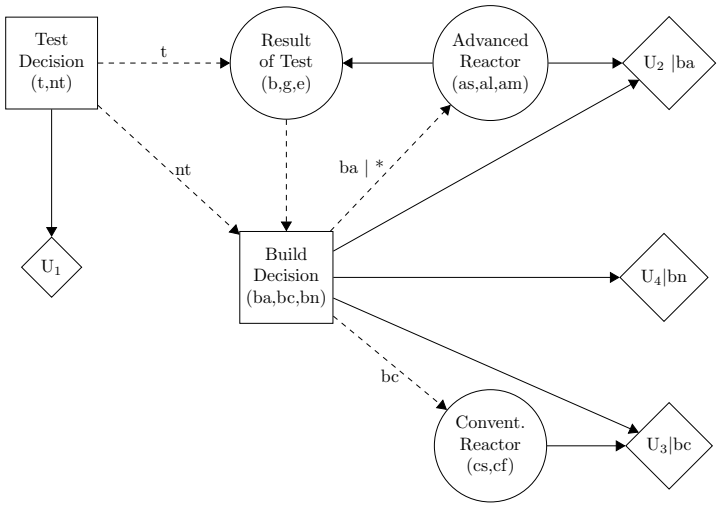


Figure 2.39: SID representation of the reactor problem

Figure 2.39 shows the SID representation for the reactor problem according to Jensen et al. (2006). The sequence of the decisions and observations is described explicit with structural arcs. Starting from the decision *Test Decision*, which does not have any incoming arcs, the distinct scenarios can be built following the structural arcs. The labels above the structural arcs describe under which conditions an arc is feasible. For example the variable *Result of Test* does not appear at the decision scenario when the *Test Decision* is not to perform the test (*nt*). In this case the *Build Decision* is the next node in the scenario. The reactor problem contains also a more complex case of structural asymmetry, namely the constraint of the test result on the *Build Decision*. This constraint is described with a composed annotation where the decision is conditioned on previous decisions and observations and appears at the structural arc between *Build Decision* and *Advanced reactor*. This arc has the annotation $ba|(T = nt \vee (T = t \wedge (R = e \vee R = g))$, which means that the decision to build an advanced reactor can only be taken when the test is not performed (T=*nt*) or the test result has been excellent or good. This annotation abbreviates *Test*

Decision as T and *Result of Test* as R. This description does not represent at the graphical level the restrictive influence of the *Result of Test* on the *Build Decision* as the annotation appears on the outgoing arc of the *Build Decision* and the guard may appear abbreviated at the graph.

The scenarios end either with the event of building an advanced reactor, building a conventional reactor or none according to the alternative choosen at *Build Decision* and indicated above the outgoing arcs of *Build Decision*. The utility nodes, which represent the benefit of constructing a certain type of reactor (U_2,U_3,U_4) are specified as fragments, where the utility value is only given for the relevant decision option ($U_2|ba, U_3|bc, U_4|bn$). This corresponds to the approach taken from AIDs and SVNs to describe explicit logically inconsistent configurations. Nevertheless the constraint between the advanced reactor's components and the *Result of Test*, namely that the combination $\{as,b\}$ is not possible is not described explicitly with a label as the property of the advanced reactor is not observed before the test result. The SID representation describes this constraint assigning a zero probability value to the incompatible combination.

Conclusion

SIDs are a very interesting formalism for representing asymmetric decision problems as they take the best features of UIDs, SVNs and AIDs and improve other aspects related to the representation of asymmetry.

SIDs admit that parts of the model have a partial temporal order of decisions (represented with cluster), a feature taken from the UID formalism. This is useful for the representation of problems where some decisions have an undefined temporal order (order asymmetry) as it avoids to represent all possible sequences of decisions and observations at the model, what obscures the structure of the problem. SIDs represent the parts of the problem having a partial order as clusters, describing concisely with structural arcs the position of the cluster in reference to the other parts of the problem. From the representation of the diabetes and dating problem we have seen how clusters are used to describe parts of the problems with order asymmetry.

Although SIDs allow parts of the model having a partial order, the overall description of the model is a total order of decisions and observations. SIDs use a SDD based graph which describes in a compact form the possible decision scenarios. This graph is formed by structural arcs which describe under which conditions a variable appears, thus giving information about the sequence of decisions and observations and the details of structural constraints. This approach is useful for the description of conditioned and disjoint scenarios, i.e., when a variable may be non-existent in certain conditions, such as we have seen at the representation of the dating problem. The decision analyst can read directly from the model under which conditions a variable appears. The approach to base the description of the asymmetric constraints only at the model which describes the chronological order can also be an inconvenient. If the variables which define a restriction are not observed successively or several variables are involved, the label above the arcs (i.e. the guard) needs to specify these conditions in the second part of the annotation. This information does not appear at the graphical level and does not describe the cause-effect relation between the involved variables. SIDs propose guards, which are a language for the description of conditions under which a variable appears taking into account previous decisions and observations as an alternative to indicator valuations and the description of constraints at the functional level of AIDs. Although the description with guards is concise and easy to understand, it is implicit to the model and does not represent explicit the restrictive influence of a variable on another. At the reactor problem we have seen an example of the description of such a complex constraint with guards, where the description of the restrictive influence of the *Result of Test* on the *Build Decision* is not obvious from the diagram.

Another important aspect of SIDs regards the representation of the uncertainty model. SIDs

use a BN based model to represent probability relationships. This implies the implicit representation of direct and conditional probabilities at the model, which are intuitive to understand and to assess from humans. This is a modification of SVNs, which represent the probability model with valuations at a separated model. Although SVNs are also able to represent conditional and direct probabilities, the representation with valuations is not that intuitive to understand, but it has the advantage that they describe clearly separated from the model of order the probabilistic relationships. Although SIDs also distinguish these descriptions, by representing conditioning arcs with solid lines and structural arcs with dashed lines, the probabilistic dependences are represented implicit in the graph. It is to see whether the combination of the model of order with the uncertainty model or whether a valuation based representation is better when the probabilistic relations involve variables which are not observed sequentially or when the problem involves a large quantity of observed variables. The SID formalism nevertheless improves an important shortcoming of the SVNs as they permit to represent unobserved variables, which are useful for the description of probabilities, but would not appear in the corresponding SVN. This difference is described at the respective representations of the dating problem.

From the representation of the three asymmetric decision problems we can conclude that SIDs are suitable both for the representation of order and structural asymmetry. SIDs combine a probabilistic model and a model for the description of information precedence and structural constraints at the same diagram, what makes them very expressive for the description and analysis of decision problems.

2.4 Conclusion of the review

In this chapter we have revised the state of art regarding the representation of decision problems with PGMs. We have seen how the description of conditional independence assumptions makes them a powerful tool for the description and analysis of decision problems. Further we have seen that the most common decision analysis models, namely IDs and DTs, are unsuitable for the representation of real-world problems with asymmetry. IDs resulted unsuitable as they need to symmetrize the problem artificially, what obscures the structure of the problem and increases the time and space requirement. DTs have almost absolute flexibility to represent order and structural asymmetry, but due to their exponential growth with the number of variables they are unable to represent large problems.

Several other formalisms were proposed which address specifically the representation of asymmetry. ExtIDs combine the representation of the uncertainty model of IDs with tree-like structures to capture the asymmetric aspects of a problem, but are not suitable for decision analysis in practice as the representation of a problem comprises lots of small diagrams. Another formalism we have seen are SVNs. This formalism combines a VN based representation of the uncertainty model with the representation of the order of decisions and observations and describes simple constraints with labels and complex constraints with indicator valuations. When SVNs were proposed, they were the most complete formalism for asymmetric decision problems at that moment, although later several formalisms for treating order asymmetry more appropriately appeared. The representation of order asymmetry was resolved best with UIDs, which are a very suitable for diagnosis and trouble-shooting problems, where the representation of order asymmetry is more important than that of structural asymmetry, which is not resolved at UIDs. Nevertheless UIDs have a major limitation: UIDs assume that a variable is observed only when all its parent decision have been made. Therefore they cannot represent problems in which a variable is observed as soon as any of its parent decision is made. This is a limitation of UIDs, which does not happen for any other formalism discussed in this chapter, because there are IDs

for which an equivalent UID does not exist. Another formalism we have seen are AIDs, which adapt the semantics of the arcs and nodes of IDs for the representation of non-sequential decisions and conditioned scenarios showing labels to describe under which condition a variable appears. Nevertheless AIDs only represent structural asymmetry efficiently but have difficulties to represent order asymmetry.

To date SIDs are the most complete framework for the representation of asymmetric decision problems, because they are suitable to represent both order and structural asymmetry. We have seen that SIDs use an ID based model to describe uncertainty combined with a compact but explicit representation of the decision scenarios, which make visible the information precedence and asymmetric constraints. SIDs combine the research of the two groups that have worked more in formalisms for asymmetric decision problems: These are Prakash Shenoy and collaborators from the School of Business of the University of Kansas (United States), which proposed SVNs, and the Machine Intelligence Group[1] at the Aalborg University (Denmark), which directed by Finn Jensen and in collaboration with Thomas Nielsen and Marta Vomlelová proposed UIDs and AIDs. As a result of this collaboration SIDs combine the best features of SVNs, AIDs and UIDs and are very suitable for the representation of asymmetric decision problems.

[1] http://www.cs.aau.dk/research/MI

3 Decision analysis networks

This chapter introduces the decision analysis network (DAN) formalism. In the first section we give a formal description of the formalism and in the second we describe how DANs represent asymmetry, illustrating the solution by means of the three asymmetric decision problems studied in the previous chapter. At the third section we present a detailed comparison of the DAN representation to the respective solutions of the alternative formalisms (i.e., the formalisms known so far) and at the fourth section we present the results of the comparison.

3.1 Decision Analysis network representation

Decision Analysis networks (DANs) are a new type of probabilistic graphical model for modeling asymmetric decision problems proposed by Díez & Luque (2010). This formalism is intended to represent order and structural asymmetry more naturally than the alternative formalisms proposed earlier.

3.1.1 Definition

Graph and variables

A DAN is an acyclic directed graph (DAG) composed of chance variables, decision variables and utility variables, with a semantic similar to that of IDs: chance variables represent real-world properties that are not under the control of the decision maker, decision variable represent the alternatives a decision maker has for a decision and utilities represent the preferences of the decision maker.

Temporal order

Unlike IDs, DANs do not require *total order* between decisions. Nevertheless if there exists a directed path from decision D_1 to decision D_2, then D_1 will be made before D_2.

Restrictions

DANs have the ability to describe *restrictions* associated to links. A restriction expresses the incompatibility between certain values of two variables connected by a directed link $X \to Y$, where X is a decision or chance node. A link restriction limits the values of Y to a subset of its domain and describes conditions which make a variable non-existent in a scenario or which make the variable appear with a subset of its values. In both cases if a variable does not exist in a given scenario it does not have any effect on other variables:

- A variable Y does not appear in a certain scenario (i.e., it is *non-existent*) if all of its values are forbidden by a state of the conditioning variable X, i.e., the allowed values of $Y' = \emptyset$. This type of restriction is termed *total restriction* and is represented with a double stripe crossing the link.

- A variable Y appears with a subset of its values in a certain scenario if a state of the conditioning variable X forbids some of its value but not all values, i.e the allowed values of Y are $Y' \subset Y$ and $Y' \neq \emptyset$. This type of restriction is termed *partial restriction* and is represented with a single stripe crossing the link.

If both types of restriction occur at a link, i.e., one value of the conditioning variable X forbids all values of Y (total restriction) and another value of X the conditioning variable forbids only some values of Y (partial restriction), this is represented as total restriction graphically describing that the variable Y does not appear in at least one scenario.

The restrictions associated to a link X $\rightarrow$ Y can be represented by means of a potential ψ defined on $X \times Y$, such that $\psi(x, y) = 1$ expresses compatibility and $\psi(x, y) = 0$ incompatibility. The meaning of a restriction potential depends on the type of the restricted variable:

- A restriction on a decision describes the available options for the different scenarios of information.
- A restriction on a chance variable indicates which outcomes of a variable are possible for the different conditioning states.
- A restriction potential on a utility describes the possibility that a decision option or a state of a chance variable has a utility value. Only possible states can have a utility value as they correspond to real states of the world. Impossible states have a special value assigned, which is an explicit description that a state never occurs.

Potentials

The probabilistic model comprises *conditional probability distributions* and *utility distributions*, which are both represented as potentials and associated to chance variables or utility nodes respectively.

A conditional probability distribution (CPT) associated to each chance node denotes the conditioning influence of the parent nodes on the chance node. In DANs the potential of a chance node Y whose parents in the graph are X, denoted by $P(y|x)$, is the conditional probability distribution of the node Y based on its parents, which takes into account the existence of restrictions between the node and any of its parents. The resulting potential $P(y|x)$ satisfies theses three conditions:

- $P(y|x)\, \epsilon\, [0, 1] \cup \{-\}$
- If there is a restriction (x_i, y) and $x^{\downarrow X_i} = x_i$ (i.e., the i-th value of the configuration x is the same as $x_i$in the restriction), then $P(y|x)$= $\{-\}$. This means that all the combinations of the potential $P(y|x)$, where X takes the value x_i are impossible, which means that the state y never occurs when the event X takes the value x_i.
- For each configuration x of X, if some of the values of $P(x|y)$ are real-numbers, then their sum is 1.

A utility distribution is associated to each utility node, where the parent nodes define the domain of the utility distribution. In DANs the potential of a utility node Y whose parents in the graph are X is denoted by $U(x)$. As in IDs, this utility distribution stores a value for each of the parent configurations. The potential $U(x)$ takes into account the existence of restrictions between the node Y and any of its parents and therefore the resulting potential $U(y|x)$ satisfies theses two conditions:

- $U(x) \epsilon \mathbb{R} \cup \{-\}$
- If there is a restriction (x_i, y), it means that certain states of x_i are not possible, and $x^{\downarrow X_i} = x_i$ (i.e., the i-th value of the configuration x is the same as x_i in the restriction), then $U(x)= \{-\}$. This value describes that certain states of the world are impossible and therefore can not have a utility value.

Representing the flow of information

DANs use two mechanisms to describe the flow of information: *always-observed variables* and *revelation arcs*.

A chance variable can be declared as *always-observed*, what means that its value is known without taking any action. Always-observed variables are depicted with a dark red colored border. Another mechanism to describe the flow of information are *revelation arcs*, which are represented graphically as links with a dark red color. A revelation arc between X and Y, where X is chance or decision variable and Y is a chance variable, means that certain values of X *reveal* the values of Y. In consequence the value of Y gets known for future decisions. The function of revelation arcs is therefore the equivalent of information arcs in IDs: they describe which variables are known for a decision, but with the subtle difference that revelation arcs denote *possible information precedence.* A revelation arc has only effect if the variable X is known and X takes any of the values which reveal Y.

A variable is known with certainty if it is always-observed, and it is (potentially) known if a revelation arc points to it. In order to analyze which variables are known for a decision D, the decision analyst must consider the variables revealed by always-observed variables and those revealed by previous decisions. As DANs allow the temporal order of decisions be unspecified, a previous decision can be a decision with an explicit temporal order or a decision with unspecified temporal order in reference to the current decision. A decision with unspecified temporal order must satisfy an additional requirement in order to reveal a variable, i.e., it must happen effectively before the current decision.

Meaning of links

In summary the links at a DAN can express different kinds of relations. The meaning of an arc $X \rightarrow Y$ depends on the type of variables it connects:

- *Causal influence*: If Y is a chance variable, the link $X \rightarrow Y$ describes *causal influence.* When the variable X does not exist in a scenario, i.e., when all its values are impossible, then Y does not exist in that scenario.
- *Functional dependence*: If Y is a utility, the link $X \rightarrow Y$ describes *functional dependence.* The variable Y does not make sense when the variable X does not exist.
- *Temporal order*: If both X and Y are decisions, the link $X \rightarrow Y$ describes the *temporal precedence* of X in reference to Y, i.e., X is made before Y.
- *Restriction*: If X is a decision or chance node, the link $X \rightarrow Y$ describes restrictions on Y imposed by X. When X exists and its value is known this restriction becomes effective at the moment of doing inference, by limiting the values of Y that can be chosen.
- *Revelation*: If Y is a chance variable and X is a decision or chance, the link $X \rightarrow Y$ describes that some values of the variable X reveal the values of the variable Y.

The first three meanings are similar to that of IDs. Revelation arcs substitute information arcs in IDs.

3.1.2 Evaluation

Díez & Luque (2010) propose as evaluation method of a DAN the transformation to an equivalent DT. The authors of the formalism mention the development of more efficient algorithms for future work (see Section 5.2 regarding future work) and remark that the purpose of the conversion to the equivalent DT is to give a clear semantics of the model.

The transformation into an equivalent DT has further the advantage that decision analysts from other fields of research can understand the proposed model of DANs better. DTs are easy to understand because they fully depict the decision scenarios and humans tend to analyze decision problems by figuring out the possible scenarios (Lacave et al., 2007). In fact in medicine for example the standard analysis tools are DTs as explained by Pauker & Wong (2005) while IDs are almost unknown.

The construction of an equivalent DT is based on the chronological order in which observations and decisions happen. The always observed variables are placed at the beginning of the tree followed by the variables they reveal. Next appear in chronological order the decisions and the variables each decision reveals. If the order between decisions is unspecified an additional node which specifies the order is necessary. The not observed variables are placed at the end of the tree. This approach yields a tree structure and the conditioned probabilities of the branches are obtained from the joint probabilities of the end scenarios using normalization.

3.2 Representation of asymmetric decision problems

3.2.1 DAN representation of the diabetes problem

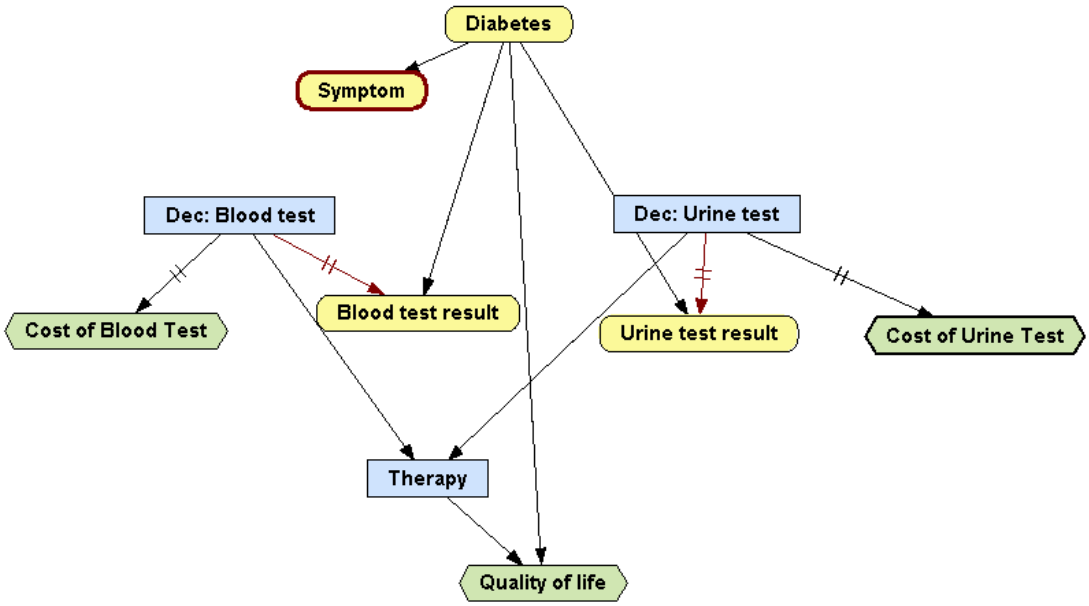


Figure 3.1: DAN representation of the diabetes problem.

Figure 3.1 shows a DAN for the diabetes problem. As DANs do not require a total order of decisions, i.e., they admit partial order, they are able to represent naturally the diabetes problem, where the order of the tests is undefined, by representing directly the test decisions and their correspondent test results in the model. This allows DANs to model efficiently order asymmetry and makes them suitable for the representation of diagnosis problems as they represent clearly the relation between the decision to make a test and the information which is revealed by this test. The diabetes problem contains a simple case of structural asymmetry. The constraints between the two tests and their respective results are described with link restrictions between the test decision nodes ant the test result node. The link restrictions establish that the decision about performing the test restricts the values of the test result. The decision of not to perform a test make all values of the test result impossible, what is a type of total restriction represented with a double stripe crossing the link. Figure 3.2a shows the GUI of OpenMarkov for the edition of link restrictions. This dialog shows that the combinations of *not test* are incompatible with any value of the test result. Compatible combinations appear in green color and incompatible combinations in red color.

Due to the occurrence of incompatibility between the test decision and the test result, some states of the test result become impossible. Figure 3.2b shows the conditioned probability table (CPT) of the test result variable. This table shows impossible states in red color.

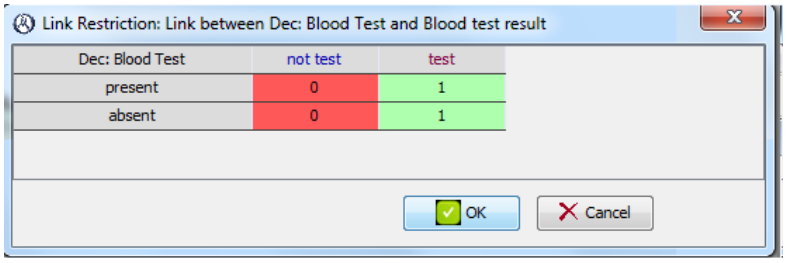
The test decisions establish also restrictions on the utility of the cost of the test. The variable *Cost of the Test* does not make sense when the decision is not to perform the test. The GUI for the edition of the restrictions on the utility values is shown in Figure 3.2c and the resulting utility values are shown in Figure 3.2d, where impossible states appear in red color.

In the formalisms that have no means to model constraints, structural constraints are represented by adding dummy states to the test result variable to express that a state is not available in certain scenarios, and negative utility values are assigned to impossible states to avoid that those states are chosen by the solution algorithm. The use of dummy states obscures the structure of the problem and augments the space and time requirements for solving the problem. In this aspect DANs have an advantage over other formalisms as they can describe constraints with link restrictions and maintain the state space of the variable small. The state space of the test result variable in DANs has a cardinality of two and makes the representation of its conditional probability table very compact (see Figure 3.2b).

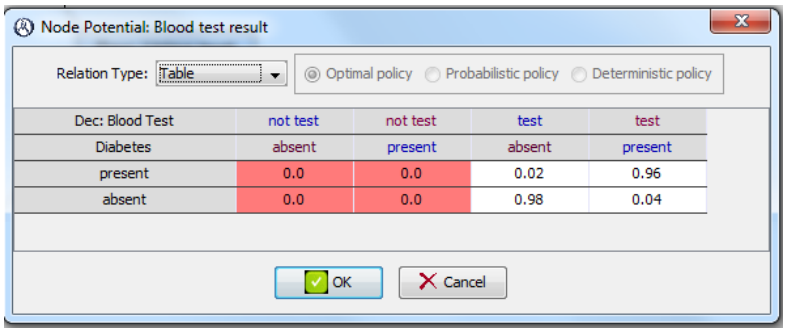
While link restrictions express constraints between variables, revelation arcs denote the flow of information. A revelation arc describes possible information precedence. The diabetes problem contains two revelation arcs between each of the test decisions and the respective test results. Figure 3.2e shows the GUI of OpenMarkov for the edition of the revealing states of the *Blood Test*. This revelation arc means that the test result becomes known when the decision is to perform the test. The test decisions are prior to the decision about the therapy (as specified with the arc of temporal precedence). So the test results are variables which are potentially known when taking the decision on the therapy. The fact that these variables are effectively known depends whether the test decision reveals the value of the test result.

As the temporal order between the two test decisions is unspecified, the informational precedence for the test decisions depends further on whether the other test happens before. For example when deciding on the *Blood Test* the *Urine Test Result* is known when the decision about the *Urine Test* has been made before and the decision was to perform the urine test. The diabetes problem shows clearly how revelation arcs describe possible information precedence. The fact whether a variable is revealed depends on the particular values the revealing variable take and the effective order in which events and decisions happen.

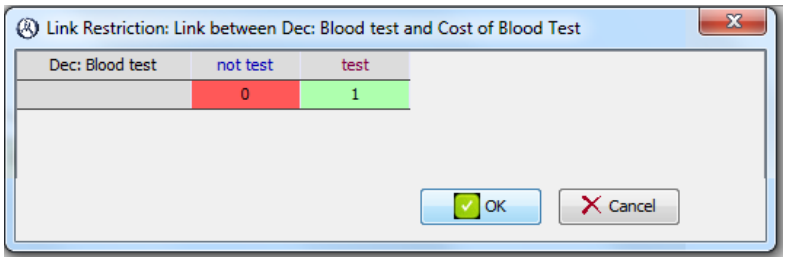
Note that link restrictions must be coherent with revelation conditions. A state of a variable cannot be a revealing condition for another variable if it restricts all values of that variable.



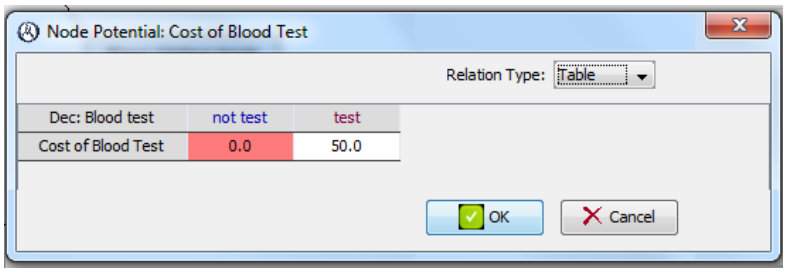
(a) Edition of compatibility values for a chance variable.



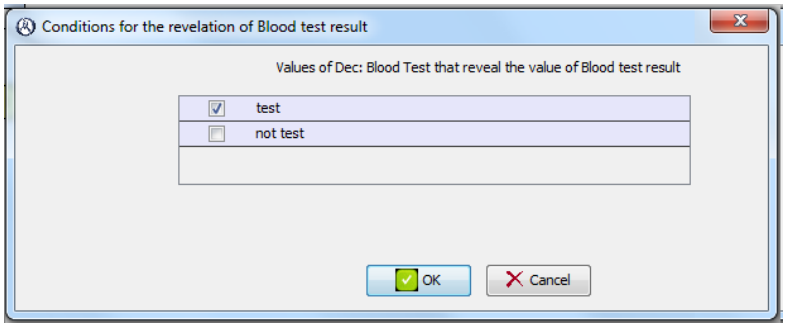
(b) Edition of a CPT.



(c) Edition of compatibility values for an utility.



(d) Edition of utility values.



(e) Edition of revelation conditions.

Figure 3.2: Edition of link restrictions and revelation conditions for the diabetes problem.

Correspondingly a state of a variable, which is a revealing condition for another variable can not restrict all of its values. The control of these conditions is not implemented currently in OpenMarkov, but it is a logical requirement of the model. In the case of the diabetes problem the variables *Test Decision* and *Test Result* satisfy these conditions. The state *not test* of the test decision is not a revealing condition as it restricts all values of the test result, and the state *test*, which is a revealing condition, does not restrict any values of the test result.

From the description of the DAN representation of the diabetes problem we have seen how link restrictions and revelation arcs are used to describe the relation between the test decision and the test result. The link restrictions describe that the test result does not exist in all scenarios and the revelation arcs describe that the test result is only known when the test is performed and gets known for future decisions. As DANs admit an unspecified order of the decisions the test decision can be represented directly and the states space of the test result variable only contain the possible states of the test result. Further information precedence is expressed with revelation arcs which are local and independent from the complexity of the problem. Collectively all these features make the DAN formalism suitable for the representation of the n-test problem.

Construction of the equivalent decision tree

This section explains the transformation of the DAN representation of the diabetes problem to its equivalent DT. The equivalent DT of the diabetes problem is shown in Figure 3.3. The always-observed variable *Symptom* (S) is placed in the first position, followed by the decisions. As the order between the tests is unspecified it is necessary to add an extra node representing the choice of order (OD) before the test decisions. Each branch contains the variable that is revealed according to the test and test decision. Finally the unobserved variable *Disease* (D) is placed at the end of the tree.

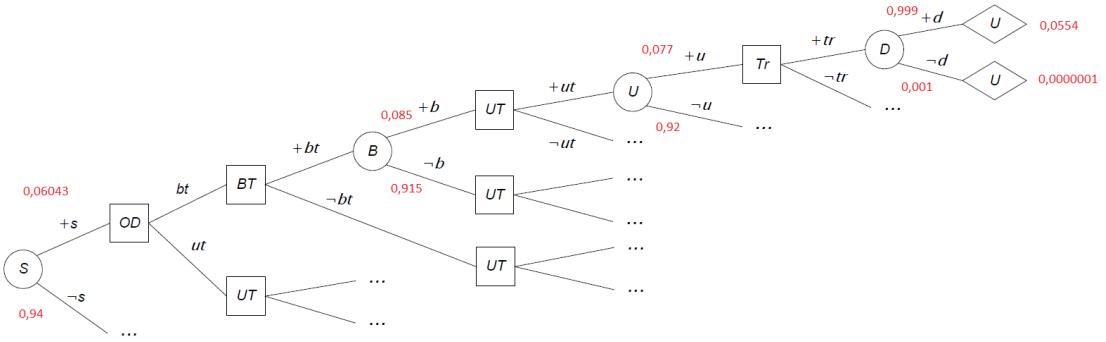


Figure 3.3: Equivalent decision tree for the DAN of the diabetes problem.

Once constructed the graphical structure it is necessary to calculate the probability of each branch in order to evaluate each branch and find the optimal strategy. The probability of a branch can be calculated from the probabilities of the scenarios using the theorem of conditional probability. The probability of a scenario is the joint probability for the values of its variables. The probability of a branch leading to a scenario is the probability of the variable of the branch conditioned on the values of the variables on the left of the branch. The appendix A.1 explains in detail the computations of the conditioned probabilities for the upper branch of the decision tree shown in Figure 3.3.

3.2.2 DAN representation of the dating problem

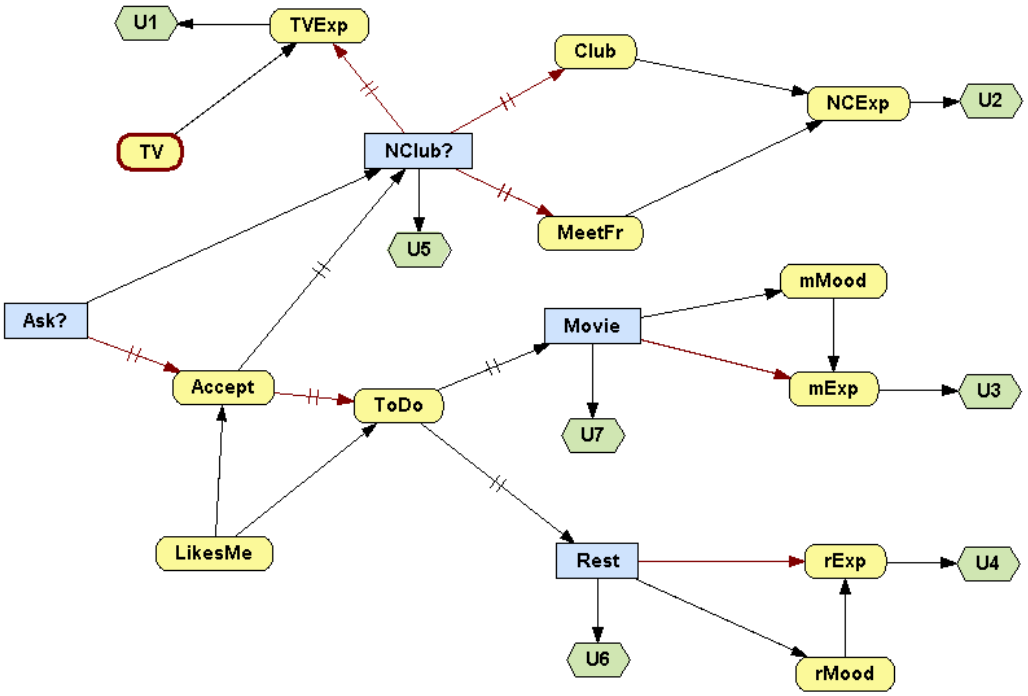


Figure 3.4: DAN representation of the dating problem.

The DAN representation of the dating problem is shown in Figure 3.4. This graph shows that the chance variable *TV* is always-observed as its value is known without taking any action. The arc between *Ask?* and *NClub?* denotes temporal precedence, so the decision *Ask?* is the first decision to take in this decision problem.

The existence of conditioned scenarios is expressed by the use of link restrictions, which are graphically depicted as double stripes crossing the link. Although the details about the restrictions are not shown with labels at the graph, the decision analyst deduces information about the existence of conditioned or even disjoint scenarios from the graph. The detailed information about the restrictions can be accessed by selecting the link and opening the link restriction table. The information flow is described with revelation arcs which are shown as links with a red dark color at the graph. Revelation arcs describe under which conditions a variable gets known. The details about which states are a revelation condition is not shown either with labels in the graph but can be easily accessed by selecting the link and going to the edition of revelation conditions. The next section explains how the DANs use revelation arcs and link restrictions to describe the asymmetry of the dating problem.

The graphical model of the DAN shows information flow by the use of revelation arcs at different times:

- The revelation arc from *Ask?* to *Accept* describes that when Joe decides to ask Emily for a date the values of the variable *Accept* are revealed.
- The revelation arc from *Accept* to *ToDo* indicates that if Emily decides to accept the date her preferences get known (either go to a restaurant or see a movie).
- The revelation arc from *Movie* to *mExp* describes that the choice for a type of movie reveals the post-movie experience of Joe
- The revelation arc from *Restaurant* to *rExp* describes that the choice for a type of restaurant reveals the post-restaurant experience of Emily.
- The revelation arc from *NClub?* to *TVExp* denotes that the values of the *TVExp* are revealed if the decision is not to go to the nightclub.
- The revelation arcs from *NClub?* to *Club* or *MeetFr* describes that these observations are made if the decision is to go to the nightclub.

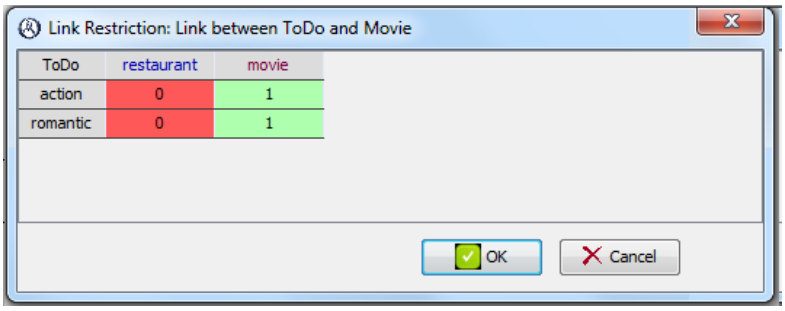
The dating problem contains very different scenarios depending on the state of different observations or decisions. DANs use link restrictions to describe constraints, which lead to situations where a given variable does not exist. The representation of the dating problem contains link restrictions in different situations:

- If Joe decides not to ask Emily for the date (*Ask?*), he will never know whether she would accept or not. The variable *Accept* is non-existent if the decision is not to ask for the date (*Ask?=no*).
- If Emily does accept the date, the variable *NClub* is non-existent, as Joe does not consider the option to go to the nightclub any more.
- If Emily does not accept the date, the variable *ToDo* is non-existent, as Joe will not ask for her preferences.
- The decision whether to watch a movie or go to a restaurant (*ToDo*) restricts the values of the subsequent decision. If Emily decides to see a movie, the choice of the restaurant does not happen. If the decision is to go to a restaurant the decision about the type of movie never occurs.
- The constraints between *NClub* and *Club* and *MeetFr* describe that these variables will not exist if the decision is to stay at home. In the same way the link restriction between *NClub* and *TVExp* denotes that the variable *TVExp* will not exist if the decision is to go to the nightclub.

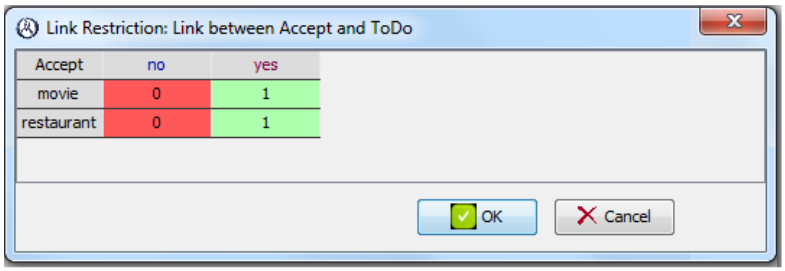
This section gives a detailed description of two of the link restrictions in order to illustrate how link restrictions are used to describe conditioned scenarios. The first example is the appearance of two disjoint scenarios as a consequence of the decision *ToDo* (see Figure 3.4). The decision *Movie* does not happen when Emily decides to go to the restaurant (*ToDo=restaurant*) and the decision of *Restaurant* does not happen when Emily decides to see a movie (*ToDo=movie*). This is described by associating link restrictions to the outgoing arcs of *ToDo*, which restrict the state space of *Movie* and *Restaurant* in each case. Figure 3.5a shows the link restriction table for the

link between *ToDo* and *Movie*. This table shows that the variable *Movie* does not exist when the decision is to go to the restaurant.

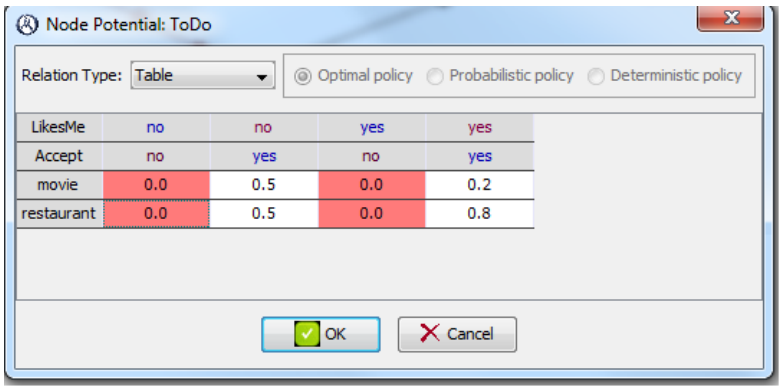
Another example is the use of a link restriction to describe the conditioning of *ToDo* on *Accept*. The observation about Emily preferences (*ToDo*) only happens when she accepts the date (*Accept=yes*). Figure 3.5b shows the corresponding link restriction table, which shows that the variable *ToDo* does not exist when the variable *Accept* has the value *no*. As a consequence to these incompatibility the CPT of the variable *ToDo* contains some impossible states (Figure 3.5c). This table depicts with red color impossible states, which have also a probability of zero.



(a) Link restriction from *ToDo* to *Movie*.



(b) Link restriction from *Accept* to *ToDo*.



(c) CPT of the variable *ToDo*.

Figure 3.5: Link restrictions of the dating problem.

The solution process of the DAN consists of the transformation to its equivalent decision tree. For the construction of the tree the decision analyst needs to know the order in which chance

variables and decisions get known. DANs use revelation arcs, temporal arcs between decisions and always-observed variables to describe the order in which observations and decisions are made. From the DAN representation of the dating problem (see Figure 3.4) we can see that the variable *TV* is initially observable. Following the flow of information described by revelation arcs it is possible to obtain an ordering of the rest of the variables. The resulting decision tree has the structure which is explained below.

At the tree first appears the variable *TV*, which is always observed. Next appears the first decision, which is the decision to ask for the date (*Ask?*) and the variables which get observed by it (*NClub?* or *Accept*). In the case Joe gets the date (*Ask=yes* and *Accept=yes*) the variable *ToDO* gets known. At this branch the next decision is to select either the movie or the restaurant which will make the variables *mMood* or *rMood* observed. In the case that Joe did not get the date (*Ask=no* or *Accept=no*), the next decision is *NClub*. If the decision is not to go to the nightclub the *TVExp* variable gets observed. If the opposite decision is taken (*Nclub=yes*), both the variables *Club* and *MeetFriends* get observed, but the order of them is not specified. The resulting decision tree is depicted in figure 2.5, where only the upper branch is shown with detail. The tree shows the asymmetry of the problem as not all variables appear in every branch.

3.2.3 DAN representation of the reactor problem

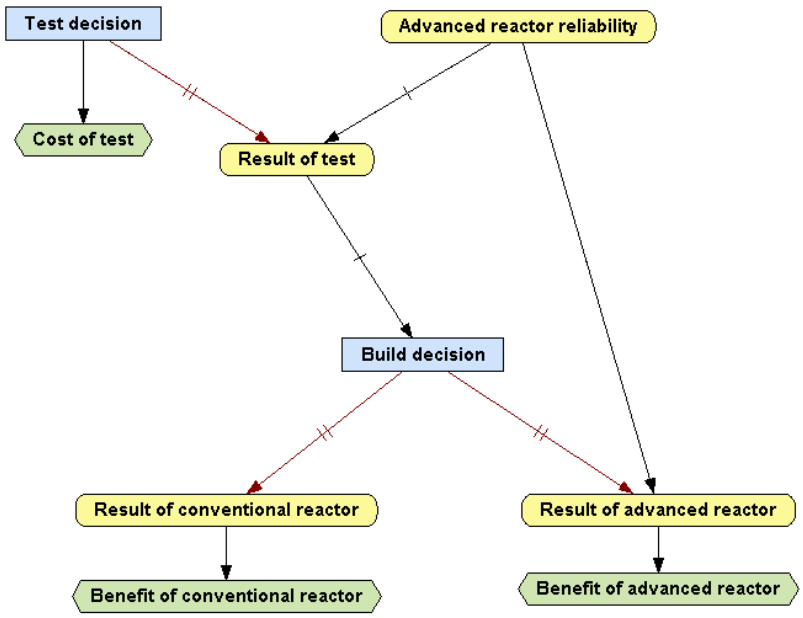


Figure 3.6: DAN representation of the reactor problem.

Figure 3.6 shows the DAN representation for the reactor problem. The node *Test decision* represents the decision to perform a test of the components of the advanced reactor. The states of *Test decision* are {*test*, *no test*}. The node *Cost of test* shows the cost of the test. The node *Result of test* represents the test results with the states {*bad*, *good*, *excellent*}, which are influenced by the reliability of an advanced reactor. When the component of the advanced reactor is *success*, the test result can not be *bad*. Additionally the test results restrict the decision on

which type of reactor to build (*Build decision*). The build decision has the options to build an advanced reactor, a conventional reactor or none. The decision to build a reactor influences the result of a conventional reactor or the result of an advanced reactor. If the test result is bad an advanced reactor is not allowed. If the test is not performed all types of reactors are allowed. The utilities associated to each type of reactor show the profit of building an advanced reactor or a conventional one for each case of success and failure. The next section explains how DANs implement the constraints of this problem.

The decision on performing the test restricts the values of the test result. The corresponding values of the link restriction are shown in Figure 3.7. This table shows that the test result does not exist when the test is not performed:

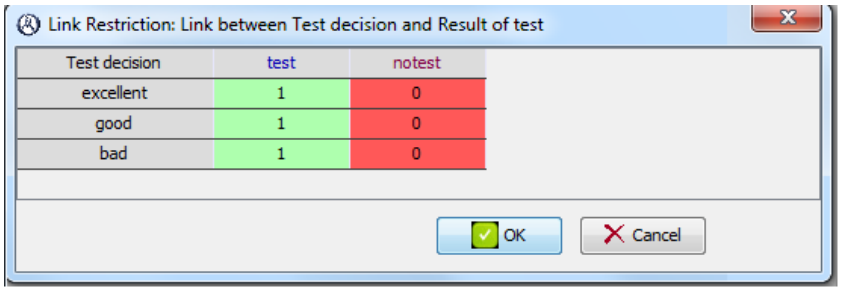


Figure 3.7: Link restriction from *Test decision* to *Result of test.*

The restriction between the *Reliability of the advanced reactor* and the *Result of test* is a partial restriction as for the value *success* the variable *Result of test* appears with a subset of its values. In the graph this restriction is represented with a single stripe crossing the link. Figure 3.8 shows the GUI for the edition of the compatibility values, which shows that the test result can not be bad when the component of the advanced reactor indicate *success*.

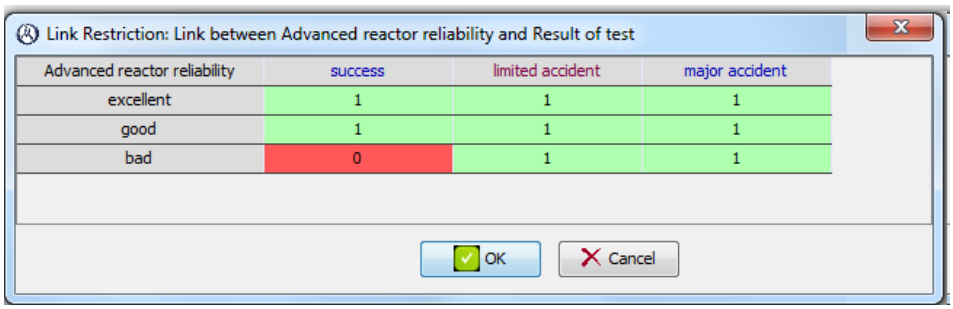
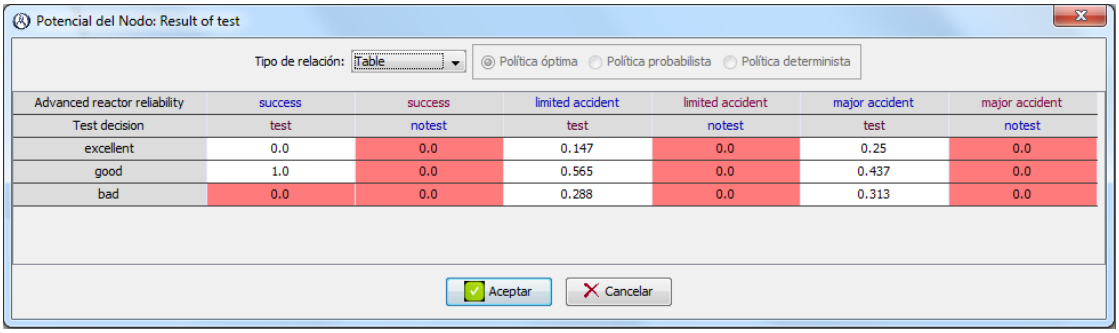


Figure 3.8: Link restriction from *Advanced reactor reliability* to *Result of test.*

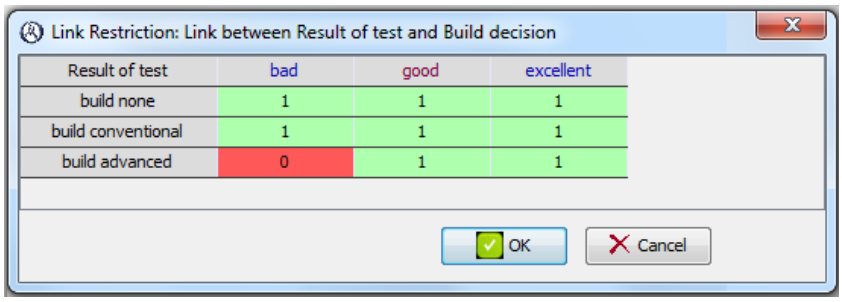
In consequence of the occurrence of the two link restrictions described before some states of the *Result of test* are impossible. The probability distribution of *Result of test* conditioned on the *Test decision* and *Advanced reactor reliability* is shown in Figure 3.9. This table shows all incompatible combinations with red color and a probability with value zero.



Advanced reactor reliability	success	success	limited accident	limited accident	major accident	major accident
Test decision	test	notest	test	notest	test	notest
excellent	0.0	0.0	0.147	0.0	0.25	0.0
good	1.0	0.0	0.565	0.0	0.437	0.0
bad	0.0	0.0	0.288	0.0	0.313	0.0

Figure 3.9: CPT of the variable *Result of test*.

Some values of the *Result of test* restrict the values of the *Build decision*. In particular when the test result is bad the option of building an advanced reactor is not available (see Figure 3.10). This type of restriction is a partial restriction as the variable *Build decision* appears for all values of *Result of test*, what is represented with a single stripe crossing the link at the graphical level.



Result of test	bad	good	excellent
build none	1	1	1
build conventional	1	1	1
build advanced	0	1	1

Figure 3.10: Link restriction between *Result of test* and *Build decision*.

The *Build decision* has a composed constraint, which means that an advanced reactor is not allowed if the test result is bad, but if the test is not performed any type of reactor can be built. Although this is not a very realistic assumption, DANs allow to model this constraint in the following way:

- The link restriction between *Result of test* and *Build decision* described before implements the restriction when the test result is available.

- If the test is not performed the *Result of test* is non-existent because of the link restriction between *Test decision* and *Result of test* described before. In this case the restrictions imposed by *Result of test* on *Build decision* are not relevant and do not influence the decision. This allows to build an advanced reactor independently from any test result as this is not available.

The link between the decision to build a reactor and the result of each type of reactor means that the event of building a given type of reactor is restricted to deciding to construct this type of reactor. Figure 3.11 shows the link restriction of the link between *Build decision* and the *Result of advanced reactor*, where only the values for the advanced reactor type are compatibles.

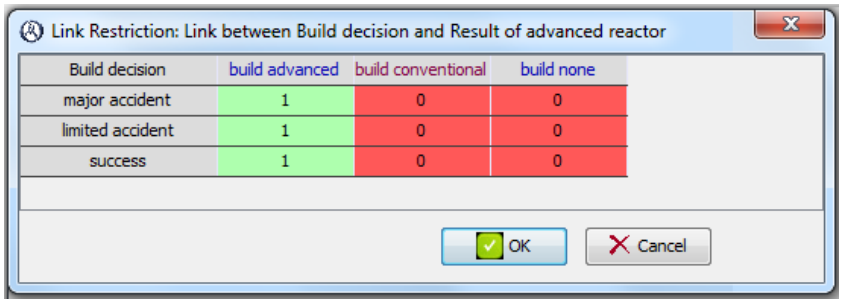


Figure 3.11: Link restriction between *Build decision* and *Result of advanced reactor*.

When the decision to build a certain type of reactor is made, the values of the result of reactor of the given type are observable. This is expressed by means of a revelation arc between *Build decision* and each of the result of reactor nodes. For the *Result of advanced reactor* the state *construct advanced reactor* is a revealing condition and for the *Result of conventional reactor* it is the state *construct conventional reactor*. Figure 3.12 shows the CPT of the variable *Result of advanced reactor*. The table shows that only the values related to the decision to build an advanced are compatible.

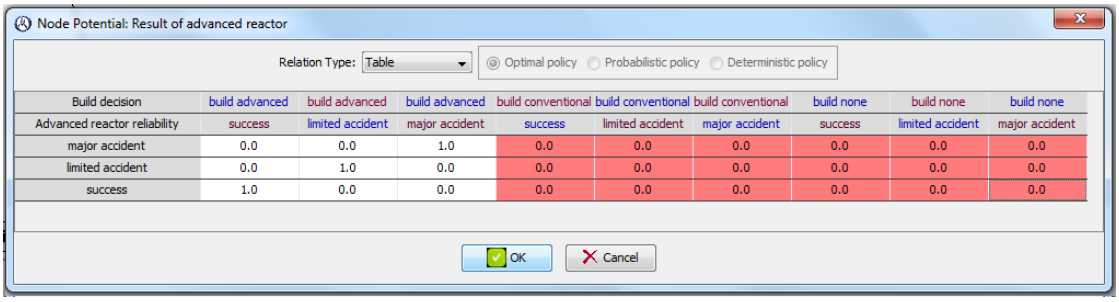


Figure 3.12: CPT for the variable *Result of advanced reactor*.

Figure 3.13 shows the utility values of the *Benefit of advanced reactor*, where only the possible states are shown.

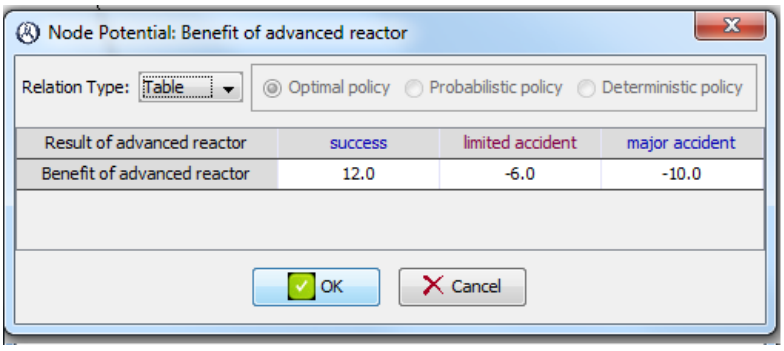


Figure 3.13: Utility values of the *Benefit of advanced reactor*.

3.2.4 Conclusion

From the representation of the three asymmetric decision problems we have seen how DANs represent different types of asymmetry efficiently. DANs provide a natural representation of both order and structural asymmetry by focusing on the representation of the underlying relations between variables using the following features:

- The temporal order between decisions can be partially defined. As the model does not require a total order, a direct representation of the decisions and the variables which are revealed from the decision is possible.

- Revelation arcs describe the information flow and under which conditions the outcome of a variable may become known. This description is natural as it represents locally which information gets revealed from certain observations and decisions, i.e., it describes the existence of different information states depending on the previous decisions or observations.

- Link restrictions represent the asymmetric nature of a problem, i.e., they describe for which conditioning states the outcomes of a chance variable are restricted and for which informational states the legitimate decision options are restricted. Link restriction further describe which states of the world are impossible by establishing link restrictions on utilities.

- The conditional independence assumptions between the variables are described by the graph as probabilistic dependences according to the underlying BN model.

At the representation of the three asymmetric decision problems we have seen how these features are used to describe the asymmetric aspects of the problems.

The representation of the diabetes problem has shown how DANs address the representation of order asymmetry. As DANs do not require a total order of decisions, i.e., they admit an undefined order, they are able to represent naturally problems where the order of the decisions is undefined by representing directly the decisions and its consequences at the model. This allows DANs to model efficiently order asymmetry and makes them suitable for the representation of diagnosis problems as they represent clearly the relation of the decision to make a test and the information revealed by this test. Another feature which makes DANs suitable for the representation of complex problems is the approach to describe information precedence with revelation arcs. Revelation arcs are a local and from the complexity independent description of information flow. At the representation of the diabetes problem we have seen how revelation arcs can be read to determine which information is known at a decision. Collectively the efficient representation of order asymmetry and the use of revelation arcs to describe information precedence gives DANs the ability to represent the n-test problem, which can not be represented with the common decision analysis formalisms.

The representation of the dating and the reactor problem has shown how DANs represent structural asymmetry. DANs describe with link restrictions the existence of structural constraints at the graphical level. At the functional level they define whether a value restricts all values of another variable (i.e., total restriction) or whether it restricts its values to a subset (i.e., partial restriction). This approach gives DANs the possibility to describe the structural asymmetry of the dating problem, which has very different scenarios depending on the previous observations and decisions. The DAN representation of this problem has shown how all the constraints of the dating problem can be represented with link restrictions, where the no-occurrence of certain decision scenarios is represented with total restrictions and at the level of a link. Further the representation of the dating problem has shown how revelation arcs are used to describe information precedence.

The DAN representation of the reactor problem was useful to describe some more specific cases of the representation of structural asymmetry. First an example was shown, where a variable restricts the decision options to a subset and how this type of asymmetry is expressed as a partial restriction at DANs (see the description of the constraint between *Result of test* and *Build decision*). Next the effect of link restrictions was described further as an example was shown where a link restriction conditioned the existence of another variable and how this affected the constraints imposed by this variable. In particular we have seen how the constraint on the *Build decision* became irrelevant in the scenarios where the test result was not available.

Summarizing, we have seen that DANs are both able to describe order and structural asymmetry. The representation of order asymmetry takes advantage of the flexibility to represent decisions with an undefined order and the local description of information precedence with revelation arcs. This makes it possible to represent the decision and the variable the decision reveals directly at the model avoiding the explicit description of all possible order of decisions and the explicit description of information precedence, what gives DANs a large degree of flexibility to represent complex problems. For instance DANs are suitable to represent the n-test problem while most of the other formalisms have difficulties to represent this problem.

Regarding the representation of structural asymmetry we have seen in the representation of all three problems how DANs use link restrictions to describe that the outcomes of a variable depend on different conditioning states and the legitimate decision options may vary depending on the informational states. Link restrictions are defined at the level of a link and describe which states of two variables are incompatible and are able to describe therefore that a variable does not appear in certain scenarios (i.e., total restriction) or that it appears with a subset of its values (i.e., partial restriction). On the other hand, revelation arcs describe the existence of different informational states and their dependence on previous observations or decisions. Additionally the description of asymmetry with DANs is local and independent of the complexity of the problem as both link restrictions and revelation arcs are defined at the level of a link. These elements are intuitive to use as the decision analyst only has to think about which combinations of states between two nodes are incompatible (link restrictions) or which values of a variable reveals the values of another variable (revelation conditions). At DANs the description of the restrictions and information flow centers on the conditioning relation between variables and not at the description of the scenario where a restriction becomes effective or an information state is used, mainly because DANs do not focus on the representation of order. This characteristic makes DANs easy to use and promising for the construction of complex problems. DANs further implement the representation of restrictions in a consistent way. The incompatible combinations are introduced at the restriction potential and both the graphical representation of the restrictions and the description of incompatible combinations of probabilities and utilities are inferred from the values of the restriction potential. This explicit description of incompatible combinations leads to a compact representation of the state space of the variables without dummy states.

3.3 Comparison with other formalisms

In the next sections we compare the capability of the alternative decision analysis formalisms with the DAN formalism regarding the representation of asymmetry.

3.3.1 Comparison with DTs

Comparing the DT representation of the three asymmetric decision problems with the respective DAN representation makes evident the main drawback of the DTs, which is the large size of the model compared to the compact model of DANs. DT depict full explicitly all possible decision scenarios, what allows them to describe every type of asymmetry. Structural asymmetry is represented by not depicting impossible scenarios and order asymmetry is modeled by expressing the different sequences of decision in separate branches. Although DT can represent asymmetry without any problem the size of its model grows exponentially with the number of variables. Therefore DT are not suitable to represent real-world problems. In contrast, DANs use a compact model capable to describe asymmetry and whose size grows linearly with the number of variables.

The ID based model of DANs has also the advantage to depict clearly the dependence and independence relationships among variables. In contrast, DTs show the variables in the order they are observed, so the dependencies relations between variables are not revealed. In this aspect DANs are a better communication tool for knowledge elicitation and communication as they describe the structure of the problem at the graphical level. Another aspect related to the representation of dependency relationships is that human experts prefer to assess probabilities in a causal direction of the variables. In this aspect DANs outperform DTs as they present direct and causal conditioned probabilities in a modular form. In contrast, DTs need the probabilities according to the order in which the variables are observed, what requires a preprocessing of probabilities for the construction and what can make it difficult to adapt the DT to changes at the decision problem. If the structure of the decision problem changes it might be necessary to redraw the whole DT, while DANs only need to update a part of the model, because the representation of probability relations is local.

3.3.2 Comparison with IDs

From the representation of the asymmetric decision problems as ID in Section 2.3.4 we concluded that IDs are not suitable for the representation of asymmetry. DANs, on the contrary are suitable for representing both order and structural asymmetry as described in Section 3.2. In the following section we analyze in detail the differences focusing on the respective solutions of the three asymmetric decision problems.

Representation of order asymmetry

A comparison of the respective representations of the diabetes problem makes evident the different approaches IDs and DAN take for the representation of order asymmetry. DANs express order asymmetry naturally by representing the test decisions and the relation to its test result directly at the model. This is possible as DANs do not require a total order of the decisions. IDs, in contrast, require a total order of the decisions, what implies that all possible decision/observations must be expressed directly at the model when the order of decisions is undefined. Therefore IDs introduce artificial nodes which represent the order of the test decisions. The representation of the test problem with IDs gets even more complex as the test result variables must contain all possible test outcomes from all test results. In the example of the diabetes problem we have

seen that with IDs the state space of the first test result has a cardinality of five, while DANs represent the test result with two states as the test result is specific for a test. This difference leads further to more complex probability potentials. For example the potential of the first test result variable of the ID has a size of 30 (see Figure 2.8) whereof most states are impossible. DANs represent each test result with a potential of four states, whereof only two states are impossible (see Figure 3.2b). The inefficiency of the representation of the two-test problem of IDs compared to the naturality of the DAN representation makes clear that IDs are unsuitable for the representation of the$n$-test problem (even for a small number of tests), while DANs are able to represent these kind of problems easily.

Representation of structural asymmetry

The dating and the reactor problems show how structural asymmetry is addressed by each formalism. A comparison of the respective ID and DAN representations shows that DANs represent the asymmetric constraints clearly and efficient, while IDs are not able to capture the asymmetric character of the problem and represent it reasonably.

IDs are not suitable for the representation of conditioned or even disjoint scenarios as they are designed for sequential and symmetric decision problems. The regularity condition of IDs requires an ordered path which contains all decisions so the underlying pattern of the ID diagram is a sequence of the decisions, what makes the representation of disjoint decision scenarios artificial. IDs represent structural asymmetry, i.e., the restriction of the outcomes of a chance variable by certain conditioning states and the restriction of the legitimate decision alternatives of a decision by different information states, only at the quantitative level. The approach IDs follow is an artificial symmetrization of the state space by introducing dummy states and assigning degenerated probabilities and utility values to impossible states. At the representation of the dating problem we have described how this approach increases the space and time required for the representation and solution of the problem and complicates the diagram with additional nodes in the domain of the utility nodes. The DAN in contrast represents structural asymmetry clearly at the graphical level, as link restrictions describe restrictions on the outcome of chance variables or on the legitimate decision alternatives. Additionally DANs do not not have the requirement for a total order of the decisions, so the diagram needs not to show the decisions as sequence, i.e., the disjoint scenarios of a problem can be described by representing the correspondent nodes graphically at different paths. For example the DAN diagram of the dating problem describes clearly that the variable *ToDo* conditions the legitimate decision alternatives of *Movie* and *Restaurant* as there is an arc describing a total restriction leading into each of the decisions (see Figure 3.4). A total restriction in DANs means that a variable does not appear for certain conditioning states and, because link restrictions can be applied on chance variables, decisions and utilities DANs are able to describe the existence of conditioned scenarios in a wide range of situations. For example DANs represent in a simple way with link restrictions the asymmetric constraints of the reactor problem on the *Build decision* and the associated utility node, which IDs model with the assignment of large negative values to utilities of the reactor problem. The use of link restrictions to describe that a value of a variable is not available in certain conditions makes the DAN representation very compact. Table 3.1 shows a comparison of the state spaces of the DAN and the ID representation. This table details for every node the states which are common for both formalisms and the name of the additional dummy state for the ID representation. The two last columns show the cardinality of the state space of the DAN and ID representation. Next Table 3.2 shows a comparison of the size of the potentials associated to a node. As the nodes of IDs have more parents and larger state spaces the potentials of the variables require more space. The table shows for each formalism the parent nodes and the size of the potential

of the node. For the DAN representation the table shows the number of possible states from the total number of states of the potential because DANs are able to describe explicitly if a state is possible. For example the variable *ToDo* has four possible states within a potential with eight states what is denoted as '4/8' (see also Figure 3.5c). This comparison confirms that the DAN representation for the dating problem is more compact than the ID representation, as the state spaces and potentials of the nodes are smaller.

A comparison of the respective representations of the reactor problem confirms further that the DAN representation is more compact and natural than the corresponding solution with IDs. This difference comes from the inefficient approach of the artificial symmetrization by IDs, while DANs can describe constraints concisely with link restrictions. For example DANs represent at the graphical level that the outcomes of the *Result of Test* and the *Build decision* are dependent on the conditioning or informational states, while IDs do not show this information at the diagram. DANs represent the variable *Result of Test* with three states while IDs need four states. This increases also the size of the potentials of the node: for example the *Result of Test* node has a potential with nine possible states at the DAN representation (see Figure 3.9) while the ID representation needs 24 states (see Figure 2.13). Link restrictions not only describe restrictions on chance variables, but also impossible states of the world and restrictions of the decision alternatives. While IDs use degenerated utility functions to describe the impossible states of the *Benefit of advanced reactor* and the restrictions of the decision alternatives of the *Build decision*, DANs use simply a link restriction to describe this information. As a consequence the DAN representation has a utility potential with three states for the *Benefit of advanced reactor* node (see Figure 3.13), while the ID representation has a utility potential with 36 states (see Figure 2.14), of which 27 are impossible states.

Conclusion

The conclusion from the comparison of the three problems is that DANs are suitable to represent both order and structural asymmetry while IDs represent them very inefficiently. Order asymmetry is represented at IDs adding artificial nodes, which represent the order of variables, whereas DANs express order asymmetry naturally representing the decisions directly. This difference comes from the fact that IDs require a total order of decisions while DANs admit an undefined order between decisions.

Next DANs also represent the structural constraints more efficiently than IDs. DANs are able to model structural asymmetry without adding dummy states by the use of link restrictions over chance variables, decisions and utility functions. Therefore the model of a DAN is more compact and computationally less expensive to solve. DANs represent with link restriction the occurrence of a restrictions at the graphical level, while IDs model constraints only at the quantitative level. This makes DANs a better communication tool as the decision analyst can recognize the existence of conditioned scenarios and constraints from the graph, while IDs do not provide this information.

Another drawback of IDs versus DANs is that they use information arcs. Information arcs make the representation of decision problems where a lot of variables are known difficult as the information precedence (at least for the next decision) is represented explicitly. Therefore problems where a lot of variables are known initially or where several variables get revealed from a decision are difficult to represent with IDs as the existence of lots of information arcs can make the diagram difficult to read. DANs use an alternative approach to describe information precedence. They use revelation arcs which describe locally and independent from the complexity of the problem which variables are known when taking a decision.

Node	Common states	Dummy state	Number of values in the DAN	Number of values in the ID
Ask?	yes,no		2	2
LikesMe	yes,no		2	2
Accept	yes, no	no response	2	3
ToDo	movie, restaurant	no preference	2	3
Restaurant	cheap, expensive	no decision	2	3
Movie	romantic, action	no decision	2	3
rMood/mMood	good, bad	unknown	2	3
rExp/mExp	good,bad	unknown	2	3
NClub?	yes, no	no decision	2	3
TVExp	good, bad	unknown	2	3
TV	good, bad		2	2
Club	good, bad	unknown	2	3
MeetFriends	yes, no	unknown	2	3
NCExp	good, bad	unknown	2	3

Table 3.1: Comparison of the size of state spaces of the variables.

Node	Parents in the DAN	Parents in the ID	Card. DAN	Card. ID
Ask?			2/2	2
LikesMe			2/2	2
Accept	Ask?, LikesMe	Ask?, LikesMe	4/8	12
ToDo	Accept, LikesMe	Ask?, Accept, LikesMe	4/8	36
rMood	Restaurant	Restaurant	4/4	9
mMood	Movie	Movie	4/4	9
rExp	rMood,Restaurant	rMood,Restaurant	8/8	27
mExp	mMood,Movie	mMood,Movie	8/8	27
TVExp	NClub?, TV	NClub?,TV	4/8	18
TV			2/2	2
Club	NClub?	NClub?	2/4	9
MeetFriend	NClub?	NClub?	2/4	9
NCExp	Club, MeetFr	Club,MeetFr	8/8	27
U_1	TvExp	TvExp	2/2	3
U_2	NCExp	NCExp	2/2	3
U_3	mExp	mExp	2/2	3
U_4	rExp	rExp	2/2	3
U_5	NClub?	NClub,Ask,Accept	2/2	18
U_6	Rest	Rest,NClub,ToDo	2/2	27
U_7	Movie	Movie,NClub,ToDo	2/2	27

Table 3.2: Comparison of the size of the potential of a node.

3.3.3 Comparison with extIDs

From the comparison of the three asymmetric decision problems we see that the two formalisms take a different approach for the representation of structural asymmetry. Regarding the representation of order asymmetry nevertheless DANs can represent this easily while extIDs inherit the shortcomings of the underlying ID model.

Representation of order asymmetry

ExtIDs are an extended ID representation having the additional ability of describing conditioned scenarios. For this reason extIDs have a lot of features in common with IDs: for example they inherit the representation of order asymmetry with artificial variables, which represent all possible sequences of observations and variables and the related shortcomings of this solution. Although the extID representation of the diabetes problem is more compact than the correspondent ID solution, the DAN representation is even more compact, because they can represent order asymmetry without introducing artificial variables which model the sequence of observations and avoid the use of dummy states. In the DAN representation a test result variable represents only the outcomes of one specific test, while in the extID representation a test result variable represents all possible outcomes from all available tests. This difference shows clearly that extIDs do not solve the problems of IDs for the representation of order asymmetry, while DANs can represent order asymmetry easily. In consequence neither IDs nor extIDs are suitable for the representation of the n-test problem, which can be represented easily with DANs.

Representation of structural asymmetry

The representation of the dating and reactor problem has shown that extIDs are able to describe structural asymmetry. ExtIDs use coalescence to describe information sharing, clipping to describe impossible conditioning scenarios, and collapsing to describe irrelevant conditioning scenarios. The solution algorithm of the extID algorithm is able to exploit these characteristics and to avoid unnecessary computations and to optimize computations. Nevertheless for large probability models an extra effort may be required to determine which distributions should be used in a distribution tree.

A main difference between extIDs and DANs is that DANs are not able to describe and exploit coalescence, while extIDs can optimize the computations by detecting states which share information. Additionally DANs do not describe the existence of irrelevant scenarios. ExtIDs use collapsed scenarios to describe the independence between variables in certain conditioning scenarios. Although extIDs are able to describe the existence of irrelevant scenarios and avoid unnecessary computations, the irrelevant conditioning scenarios which are explicitly described at extIDs often happen in consequence to the representation of irrelevant information in the diagram. The underlying model of extIDs are IDs, which have dummy states and have a more complex graph than DANs as they have information arcs and additional arcs leading to utilities to describe restrictions on decisions, what makes some conditioning scenarios more complex. See the comparison of state space and number of predecessors between DANs and IDs in Table 3.1 and 3.2 to check that an ID representation is more complex.

DANs and extIDs are both able to describe the occurrence of impossible scenarios. DANs use link restrictions while extIDs use clipped scenarios. The solution of DANs is more efficient as they avoid the existence of dummy states, while extIDs still have dummy states at the functional and numerical level, although they describe with distribution trees that these states are impossible.

An import difference between DANs and extIDs is that the latter do not show the occurrence of asymmetry constraints at the graphical level. DANs use a visual description of the occurrence

of constraints at the graphical level, but extIDs represent constraints only at the functional level with conditioning functions. Although the description of the asymmetric constraints at extIDs is fully exhaustive and more detailed than at DANs, they use a description at two levels with a lot of small tree-like diagrams to fully describe the problem. This can make the automatic representation of a problem difficult because several models may be involved.

3.3.4 Comparison with SVNs

SVNs are a hybrid of valuation networks and sequential decision diagrams and are structurally different from DANs, which have an ID based model. Beside from the structural difference which concerns the flexibility to represent certain types of probabilistic models, SVNs are equally suitable to represent structural asymmetry than DANs. At the contrary and related to the sequential decision diagram (SDD) based component of SVNs, SVNs have difficulties to model order asymmetry, which can be represented with DANs easily. A comparison of the three asymmetric decision problems shows these differences with detail.

Representation of order asymmetry

An analysis of the respective representations of the diabetes problem shows that DANs represent decision problems with order asymmetry more efficiently than SVNs. SVNs require a total ordering of the decisions and observations and therefore must include all admissible decision and observation sequences at the model, while DANs admit partial order. The consequence is that DANs represent the test decisions directly, while SVNs introduce artificial variables which represent the order of the tests. From the respective representations of the diabetes problem another important difference becomes evident. SVNs use information arcs, what makes the representation of complex decision problems with SVN difficult and the representation of the n-test problem infeasible. At the contrast DANs substitute information arcs with revelation arcs, which describe locally and independent from the complexity of the problem information precedence. Both the ability to represent order asymmetry naturally and the use of revelation arcs instead of information arcs make DANs suitable for the representation of the n-test problem.

Representation of structural asymmetry

From the representation of the dating and reactor problem we have seen how each formalism addresses the representation of structural asymmetry. Both formalisms avoid the use of dummy states by describing constraints explicitly. DANs describe the existence of impossible scenarios with link restrictions at the level of the link. The edition of link restrictions is intuitive as the decision analyst only has to think about the states which are incompatible between two variables or decisions. SVNs instead model constraints using labels or indicator valuations. Labels above arcs are used to describe simple constraints involving two variables and indicator valuations describe complex constraints, which can be partial restrictions or constraints which involve several variables. Valuation indicators can get quite complex as they can be associated to several variables, but they have absolute flexibility in specifying any constraint and therefore can describe more specific constraints than DANs. The approach of indicator valuations to encode constraints is to number out all compatible states, what makes them more difficult to understand for a human decision analyst. DANs describe constraints following the opposite approach: they specify the incompatible combinations between two variables at the level of a link. Additionally the description of constraints as labels follows a different approach. The decision analyst describes the restrictions indicating when they take effect, i.e., when the occurrence of a variable is restricted. This can be difficult for the decision analyst when building the model of

order (i.e., the SDD based sub graph) as he has to have in mind all the previous observations and decisions which might cause restrictions.

Another aspect related to the representation of constraints is the use of valuation fragments in SVNs. Valuation fragments are an efficient representation of the probabilistic model as they describe incompatible combinations explicitly. The use of fragments requires coherence between the definition of constraints (labels or indicator valuations) and the definition of the incompatible combinations as fragments. DANs avoid to control coherence between the definition of constraints and the probabilistic model as the relation between the restriction potential (definition of constraints) and the probability or utility potential is defined (see the description of the potentials in the Section 3.1.1). Therefore the incompatible combinations are automatically inferred from the definition of the constraints.

SVNs and DANs differ in the representation of constraints at the graphical level. SVNs describe on one hand the conditions under which a decision scenario appears with labels at the model of order, i.e., they describe constraints when they become effective and, on the other hand, they use indicator valuations to describe more complex constraints, which can not be represented with labels at the model of order. Indicator valuations describe the cause-effect relation between the variables, which form the restrictions. In this aspect, indicator valuations are similar to link restrictions because both describe the cause of the restriction at the graphical level, but with the difference that link restrictions only involve two variables and indicator valuations may involve more variables. Further indicator valuations and link restrictions only describe the existence of the restriction, but do not show details about their values at the graphical level. DANs further distinguish graphically whether the restriction is total or partial, an information which SVNs do not represent at the graphical level. We have seen that the SVN representation of the dating problem describes at the graphical level concisely under which conditions each scenario is possible as this problem only comprises simple constraints, which can be described with labels above the arcs. At the contrast the constraints of the reactor problem are more complex and can not be represented with labels. We have seen how indicator valuations are used then to express these constraints, while the DAN representation used link restrictions to describe these constraints providing a simpler representation.

Another difference between DANs and SVNs is the description of the probabilistic model. DANs represent direct and conditioned probabilities at the probabilistic model, a representation that is easy to understand for humans. For example in the case of the diabetes problem a representation with conditioned probabilities is convenient as diagnosis problems are built usually with direct probabilities such as the sensitivity and specificity of a test and the prevalence of a disease. Nevertheless the requirement of conditioned probabilities might imply a preprocessing when probabilities are not available in a direct or conditioned form. SVNs have weaker requirements for the construction of the probability model. The probability model of SVNs is constituted by probability valuations, which are multiplicative factors of the joint probability distribution. These probabilities are not necessarily conditioned probabilities, what gives SVNs more flexibility at the description of the probabilistic model, especially when the model is built from data and the conditional form is not available. Both DANs and SVNs use a modular description of the probability model and are therefore easily adaptable to changes at the decision problem. The representation of SVNs is even more modular as valuations are modeled explicitly apart from the model of order.

When a decision problem does not have order asymmetry, the approach of SVNs to describe explicitly the sequence of decisions and variables results useful because the decision analyst can explore all possible scenarios. DANs only specify information flow up to a partial order as they admit undefined order between decisions and thus an explicit representation of all sequences at the graphical level would not make sense. A drawback of the explicit representation of sequences

at SVNs nevertheless is that unobserved chance variables might not be represented. These are variables which are not observed in any decision scenario nor appear in the domain of an utility function. As including unobserved variables in a decision tree like graph makes its structure more complex, the alternative is to marginalize these variables out of the probability model, what involves a pre-processing of the probability model. Therefore SVNs usually do not represent unobserved variables. In contrary, DANs can represent unobserved variables without any problem. The representation of unobserved variables in the model is useful as using auxiliary variables makes it easier for the modeler to describe the probability model. Another drawback of the explicit description of the order of variables is that the resulting diagram can become complex when a lot of variables are known initially or revealed from a decision. As the model of order represents information precedence, the order in which the chance variables are observed between two decisions is represented explicitly, although this information is irrelevant for the analysis of the decision problem. The diagram of a DAN in contrast remains simple as DANs describe information precedence with revelation arcs, what is a local and from the complexity of the problem independent description.

Conclusion

In summary, we can conclude that both SVNs and DANs are suitable to represent structural asymmetry, although the description of constraints with DANs might be more intuitive. We have also seen that only DANs have resulted suitable for the representation of order asymmetry. SVN diagrams show both a model for the order of decisions and observations and a model for the probability relationships in the same diagram, while DANs only describe the latter. This is not a drawback of DANs as the explicit representation of the order of decisions and observations implies unsuitability to represent order asymmetry. For this reason SVNs are suitable for the representation of decision problems with structural asymmetry but not for order asymmetry.

3.3.5 Comparison with AIDs

AIDs are a special type of ID for the representation of asymmetric decision problems which revise the inflexibility of IDs to model asymmetry by dropping the rigid requirement for a total order of the decisions. They adapt the semantics of the components of the ID to describe that the value of a variable may be conditioned on previous observations or decisions. For this reason AIDs are similar to DANs as both are based on IDs and have elements that describe asymmetry. From the respective representation of the three asymmetric decision problems we have seen that both formalisms are able to represent structural asymmetry, but that order asymmetry can only be represented efficiently with DANs. A comparison of the solutions of each formalisms shows the differences which make AIDs not suitable for the representation of order asymmetry and the minor differences between the two formalisms regarding the representation of structural asymmetry.

Representation of order asymmetry

AIDs can represent decision problems with a partial order only when the decisions are totally independent, i.e., when the decisions happen in disjoint scenarios. In the case of the diabetes problem, a representation with partial order is not possible, because the test decisions are not totally independent and the information precedence is described explicitly with information arcs. AIDs therefore must represent the diabetes problem with a total order of the decisions what implies the representation of the order of tests with artificial nodes. DANs in contrast admit the representation of decisions with undefined order and therefore represent the order asymmetry

naturally as the test decision and the related test results are represented directly. Additionally the use of informational arcs of AIDs make the diagram of the diabetes more complex than the DAN representation. DANs describe information precedence with revelation arcs, which describe locally and independently from the complexity of the problem which variable is known when making a decision. Therefore the DAN representation is suitable for the representation of the diabetes and the n-test problem, while the diabetes problem representation of AIDs is complex and the representation of the n-test problem unfeasible.

Representation of structural asymmetry

Both DANs and AIDs are suitable for the representation of structural asymmetry as we can see at the representation of the dating and reactor problem. As both can describe constraints none of them needs to use dummy states, although the approach to describe constraints is different. DANs have a unique approach to represent the fact that a variable restricts the values of another variable: they store incompatible combinations of states between the variables of a link in a restriction potential; when the restrictions are changed by the user, the underlying probability model is updated. AIDs represent structural and functional asymmetry at two different levels: they represent the occurrence of a variable with labels at the qualitative level and specify the details about the restricted values at the quantitative level. The functional asymmetry of AIDs, where a variable restricts the values of another variable to a subset, is specified at the quantitative level and only if this restriction is on a decision it is represented at the graphical level (with a restrictional arc). If a functional restriction affects a chance variable, the constraint is not represented at the graphical level. DANs in contrast describe the existence of constraints at the graphical level with link restrictions and they depict separately if the constraint makes a variable impossible in a scenario (total restriction) or whether it restricts its value to a subset (partial restriction). Although DANs do not show labels at the graphical level to describe the conditions for the occurrence of a variable, DANs describe the existence of both types of asymmetry of AIDs and in a wider range of situations. Details about these differences will be given in the subsequent discussion of the dating and reactor problem representation.

The first difference regards the use of labels at the graphical level. AIDs use labels at the qualitative level to describe under which conditions a variable occurs, while DANs hide the details of the specification of constraints. The description with labels is useful as the dating problem contains very different conditioned scenarios and so the decision analyst can read directly form the graphical model under which conditions a variable appears. For example the labels make visible that the decision *NClub* never occurs at the same scenario with *Movie* or *Restaurant*. The use of labels at the graphical level is subtle as it can make the model unreadable when the state spaces of the variables are large and imply a lot of labels. This difficulty does not appear at the representation of the dating problem because the state spaces of the variables are small.

DANs and AIDs are both able to describe constraints, but DANs can describe some constraints in a more straightforward way, as link restrictions are applicable in a wide range of situations. Constraints are represented with restrictional arcs at AIDs and link restrictions at DANs. Both have the same meaning and describe for which configuration of the parent node some states of the second node are impossible. But restrictional arcs from the AID formalism can only describe restrictions on decisions, while link restrictions can describe restrictions on decisions, chance variables and utilities. In the situations where restrictional arcs are not sufficient to describe the constraints, AIDs combine labels of the qualitative level with information of the quantitative level to describe under which conditions a variable occurs. The following paragraphs describe this difference by illustrating them with some examples.

The occurrence of a constraint between a chance variable and a decision is represented

very similar with link restrictions or a restrictional arcs. For example the constraint between *ToDo=movie* and *Movie* is described at AIDs with a restrictional arc and a restrictive function (see Figure 2.30). The same constraint is described with a link restriction at the DAN formalism (see Figure 3.5a). Both representations describe at the graphical level the existence of the constraint as restrictional arcs and link restrictions have a different appearance. AIDs use a restrictive function and DANs use a table of compatible states to describe the admissible states, which contain in essence the same information.

On the other hand a constraint between two chance variables cannot be modeled with a restrictional arc while DANs can represent it with a link restriction. For instance the constraint between *Accept* and *ToDo* specifies that the event of not accepting the invitation makes the values of *ToDo* impossible in some scenarios. DANs express this constraint by means of a link restriction, where all values of *ToDo* are impossible for the state *Accept=no*. See Figure 3.5b for the edition of the compatibility values of this constraint and figure 3.5c for the resulting CPT of *ToDo*. AIDs cannot model this constraint between *Accept* and *ToDo* in such a simple way as DANs, because the constraint appears between two chance nodes and hence restrictional arcs cannot be used. Additionally as AIDs distinguish structural and functional asymmetry, this constraint is represented in two levels, the qualitative and quantitative level. At the qualitative level it is necessary to attach a label to the node *ToDo*, which specifies that this node only occurs if the condition *Accept=yes* is satisfied. At the quantitative level it is necessary to assign undefined values to all states of the partial probability potential, which are conditioned on the state *Accept=no*. The corresponding partial probability potential is shown in Figure 2.29.

From the respective representations of the reactor problem we can see an important important difference regarding the description of constraints. The restriction between the property of the advanced reactor and the test result, namely that the test result can not be bad if the advanced reactor property indicates success, is a functional constraint about a chance variable (it describes a restriction on the possible outcomes of a chance variable). AIDs describe this constraint only at the quantitative level (see the partial probability potential shown in Table 2.8) and not at the qualitative level. DANs in contrast indicate clearly its occurrence at the graphical level describing a partial restriction (see Figure 3.6).

Another difference between AIDs and DANs regards the description of incompatible combinations at utility nodes. AIDs do not explicit describe incompatible combinations at utility nodes as they assign zero values to the correspondent combinations of the partial utility potentials at the quantitative level. For instance the utility potential of the *Benefit of advanced reactor* contains zero values for the combinations corresponding to the decision to build a conventional reactor or none. The assignment of zero values nevertheless can introduce errors when the problem contains negative utilities, as happens at the reactor problem. Although AIDs solve this problem with an artificial transformation of the utility values prior to the evaluation, DANs avoid this problem as they describe explicitly incompatible combinations with link restrictions.

In reference to the specification of constraints we can summarize that DANs have a more simple and broader approach by using link restrictions. In some cases AIDs cannot use restrictional arcs to express constraints. In these cases they must use descriptions at two different levels. They use labels at the qualitative level, which specify conditions and introduce undefined values into the partial probability table, which must be in concordance with the conditions of the labels at the qualitative level. In contrast link restrictions are specified in a unique step by the definition of incompatible combinations at the restriction potential, which are propagated automatically to the probabilistic model (see Section 3.1.1). This avoids an additional control of consistence between different levels of the model as the probabilistic model is updated by the values of the restriction potential and the graphical level represents the values of the restriction potential. At the representation of the asymmetric problems we have seen several times how the definition

of an incompatible combination at the restriction potential is propagated to the probabilistic model and represented graphically. In particular the probability potential of the second node is updated to reflect impossible states with zero probability and non-editable positions.

Regarding the definition of incompatible combinations, partial potentials or functions of AIDs are similar to the restriction potentials of DANs. Both specify undefined or impossible states, but DANs define the link restrictions at the level of link between two variables, while AIDs only define partial functions at the level of a link. Partial probability or utility potentials are defined at the scope of the second variable, what may be more complex as more variables can have a conditioning influence on the variable.

Comparing the information flow of AIDs and DANs we can state that test-arcs have a similar meaning as revelation arcs. In AIDs a test-arc between a test-decision node X and a chance variable Y describes that the variable X may be observed depending on the values of the decision. DANs describe the same situation with revelation arcs. A revelation arc between node X and Y describes that Y is observed for some values of X. DANs are more flexible as they can describe that a decision reveals the value of a chance variable, but also that a fortuitous event — a chance variable — reveals the state of another chance variable. The following section describes the differences between AIDs and DANs regarding the description of information flow illustrating them with some examples from the dating problem.

The information flow between a decision and a chance variable can be modeled with a test-arc or revelation arc in each formalism. The meaning of both arcs is the same, the decision reveals the state of the chance variable. For instance the test-arc between *Ask?* and *Accept* describes that the state of *Ask?* determines whether or not *Accept* is potentially observed. The graphical representation is also similar, as test arcs appear in combination with test-decision nodes (represented as triangles) and DANs depict revelation arcs with a different color. AIDs use labels to describe for which decision option the chance variable gets observed, while DANs do not show labels but store the states which are revealing conditions with the link.

Information flow can also happen between two chance variables, for example when decisions corresponding to other persons appear at the problem. For example the flow of information in the dating problem describes that the decision of Emily to accept the date reveals potentially the value of the variable *ToDo*. This is easy to model with a revelation arc but AIDs do not describe this flow of information. The explanation for this difference lies in the purpose of revelation arcs. DANs use revelation arcs as substitutes of information arcs to describe which variables are known for a decision. The description of information flow with revelation arcs is local and independent from the complexity of the problem. Information arcs at the opposite are not a local description and the explicit description of information precedence might make the diagram complex. For example the AID diagram contains the additional information arc from *Accept* to the decisions *Movie* and *Restaurant*. Although these arc describe a possible precedence and have labels which describe under which conditions they are applicable, the resulting graph is more complex.

Conclusion

We have seen that order asymmetry can only be represented efficiently with DANs, as AIDs represent decisions which are not completely independent with a total order. On the other hand, both formalisms are suitable for the representation of structural asymmetry, although there are some minor differences in the way constraints are described. First of all, both formalisms describe structural constraints explicitly and thus avoid the use of dummy states. Nevertheless we have seen that DANs describe the occurrence of both types of asymmetry (structural and functional) at the graphical level and in a wider range of situations than AIDs. This happens as restrictional arcs from the AID formalism can only describe restrictions on decisions, while DANs are able to describe constraints on decision, chance variables and utilities. From the representation of the dating problem we have seen that in some cases AIDs cannot use restrictional arcs to express constraints, which are described with link restrictions in a simple way. In these cases AIDs use descriptions at two different levels. They use labels at the qualitative level, which specify conditions and introduce undefined values into the partial probability or utility potentials, which must be in concordance with the conditions of the labels at the qualitative level. DANs avoid the specification of restrictions at different levels as there is a direct correspondence between the restriction potential, the underlying probabilistic model and the graphical representation. Another difference regards the scope of definition of constraints. DANs specify incompatible combinations always at the level of a link, thus involving two variables, while AIDs specify restrictions on chance variables (partial potentials) at the level of a variable, what can be more complex as more variables can be involved.

Comparing the information flow of AIDs and DANs we can state that test-arcs have a similar meaning as revelation arcs. In AIDs test-decisions and test-arcs describe the decision to look for more evidence, what is useful for the description of information gathering patterns or diagnosis problems, where the acquirement of information has a cost and therefore a decision to decide upon assuming this cost must be included in the model. As explained at the representation of the diabetes problem, test-arcs are in principle suitable to model a test problem, but they are unsatisfactory when the decision is not binary, namely when the decision has several options a label is necessary to specify to which decision the chance variable corresponds. In contrast revelation arcs are able to describe the eventual observation of a variable independently of the number of decision alternatives. Additionally the description of information flow in general is different in DANs than in AIDs. DANs use revelation arcs to describe locally and with independence from the complexity of the problem which variables are known for a decision. A revelation arc can describe that a decision reveals the value of a chance variable, but also that a fortuitous event — a chance variable — reveals the state of another chance variable. The second type of information flow is not described at AIDs as they use information arcs to describe (possible) information precedence. This approach is not independent from the complexity of the problem and might make the representation of larger problems complex.

An important difference between AIDs and DANs is the use of labels at the graphical level. AIDs describe with labels under which condition a variable appears and DANs only describe the occurrence of constraints. The approach of AIDs is useful as the decision analyst can read information about asymmetry directly from the graphical model and break the original problem in smaller sub-problems, so that the decision analyst can focus on a specific decision or see the different decision scenarios, what improves the understanding of the decision problem especially when the problem is complex. Nevertheless the representation of labels at the graph can make the diagram difficult ot read when the problem is complex, especially when the nodes have large state spaces. For this type of problems DANs are more suitable, as they tell apart the occurrence of constraints from their specification.

3.3.6 Comparison with UIDs

From the representation of the three asymmetric decision problems we see that both UIDs and DANs are able to represent order asymmetry while only DANs result suitable for the representation of structural asymmetry.

Representation of order asymmetry

DANs and UIDs have the common characteristic that they do not have the requirement for a total order of decisions, so they can represent efficient decision problems with order asymmetry. Diagnosis problems are represented in a compact form showing the test decisions directly. Both formalism do not have informational arcs and therefore are suitable to represent diagnosis problems with several tests. Whereas the representation of the n-test problem is unfeasible with other formalisms which require a total ordering of decisions, DANs and UIDs are capable to represent it in a very compact form. Nevertheless the representation of a diagnosis problem with UIDs is a little bit larger, because UIDs can not describe explicitly constraints, so they model the fact that the test result is not available with a dummy state. DANs in contrast describe this constraint with link restrictions and represent the test results variables with a smaller state space.

Another relevant difference is that with DANs a chance variable is observable when one of its preceding decisions is made while with UIDs it is necessary that all of its preceding decisions are taken. This gives an advantage to DANs over UIDs as they can represent problems where a chance variable is revealed by any of two decisions, although at the representation of the three asymmetric decision problems we have not seen that difference.

Representation of structural asymmetry

The main drawback of UIDs versus DANs is that UIDs can not describe structural constraints explicitly. The existence of constraints is described with dummy states and therefore the information of asymmetry is hidden at the specification of the state space of the variables and the potentials. This has the inconvenient of obscuring the structure of the problem and increasing the space and time requirement to solve the problem. For this reason UIDs are unsuitable for representing structural asymmetry as we have described at the representation of the dating and reactor problem.

Conclusion

DANs and UIDs use both an alternative approach to information arcs for the description of information precedence. This gives them the ability to represent complex problems such as the n-test problem, where the representation of order asymmetry is an issue. Nevertheless UIDs have a limitation: they can not represent problems where a variable is revealed by one of its preceding decisions, while DANs can represent this kind of problems. Additionally UIDs are less complete than DANs because they do not describe structural constraints explicitly, what makes them less appropriate for the representation of decision problems with structural asymmetry.

3.3.7 Comparison with SIDs

SIDs provide both solutions for the representation of order and structural asymmetry as this formalism improves features from SVNs, UIDs and AIDs. They differ from DANs as they use informational arcs and include a model of the order of observations and decisions, but compare equally with DANs in almost all other aspects.

Representation of order asymmetry

SIDs take the same approach as DANs and UIDs for the representation of order asymmetry. If the order of the decisions is undefined at the problem, they represent them with partial order at the model. SIDs admit parts of the model to have partial order, what is represented graphically as a cluster. DANs do not describe explicitly which parts of the model have an undefined temporal order, but this becomes evident if the temporal order between decisions is not defined. In consequence both formalisms describe the order asymmetry of the diabetes problem representing directly the test decisions and its respective test results. Nevertheless for complex problems, such as the n-test problem, the representation with a cluster is more efficient than the respective representation with DANs. The n-test problems describes that n tests can be ordered before taking the decision about the therapy. In this case DANs need to draw a temporal arc from each *Test Decision* to the node *Therapy*, while SIDs represent the n test decisions and the respective test results in a cluster, having this cluster a single outgoing arc which describes the temporal precedence of the tests in reference to the decision about the therapy. We see that in this case the SID representation is more efficient as it represents the information precedence easier. In general, the SID representation of the n-test problem avoids to represent $n-1$ arcs. We conclude that both SIDs and DANs are suitable for the representation of diagnosis problem and that both are able to represent the n-test problem, although the SID representation is more efficient.

Another relevant difference is related to the description of information precedence. SIDs can either describe explicit information precedence with informational arcs or use the approach of UIDs, when the decisions have an undefined temporal order, and represent them inside a cluster. In the latter case, a chance variable is revealed when all its predecessor decisions are made (such as happens at UIDs). This can be a limitation of SIDs for the representation of decision problems, where a chance variable is revealed by any of two decisions. We think that this kind of problem can not be represented at least with an unspecified temporal order inside a cluster and it would be interesting to analyze how SIDs finally solve this problem. In contrast DANs have not difficulty to represent this kind of problems because a chance variable is revealed by any of two decisions.

Representation of structural asymmetry

The comparison of the dating and the reactor problems representation shows how each formalism addresses structural asymmetry. Although the approach to describe structural asymmetry is different — SIDs describe at the model of order under which conditions a variable appears, i.e., when the constraint becomes effective, and DANs describe the cause of the restrictions, i.e., the incompatibility between variables — both are able to express the constraints explicitly and they avoid the use of dummy states. From the representation of the dating problem we can see how DANs use link restrictions to describe the restrictions between variables, while SIDs use annotations on the structural arcs to describe the conditions under which a variable appears. The conditioned and disjoint scenarios of the dating problem are expressed as total restriction in DANs and with simple labels in the SID formalism. The use of labels makes the SID diagram very explanatory as it describes at the graphical level the possible decision scenarios and details under

which conditions they occur. For example the decision analyst can read from the model that the decision *Movie* does not appear when Joe decides not to ask for the date (Ask=no). DANs instead describe at the graphical level the occurrence and the relation between the variables which cause the constraint.

Another difference visible from the representation of the dating problem is that DANs only describe the information flow up to a partial order, while SIDs have an explicit description of the order of observations and decisions. DANs and SIDs have in common that both represent unobserved variables and their influences on other variables in the model. For DANs the representation of unobserved variables is not a problem, because the diagram does not comprise an explicit representation of the sequence. In contrast SIDs improved the representation of the model of order (a feature taken from SVNs) to include this type of variables.

A comparison of the reactor problem representations shows how SIDs use annotations on structural arcs to describe complex constraints. SIDs provide a language (named guards) to describe the constraints with a special syntax. This language allows to define for which value of the current variable an arc is feasible, but also whether it depends on previous decisions and observations. Therefore SIDs have the ability to describe complex constraints, as we can see from the description of the composed constraint between the *Result of Test* and the *Build Decision*, which is described with the following annotation, $BD = ba|(T = nt \vee (T = t \wedge (R = e \vee R = g)))$. Although the above mentioned constraint involves three variables (*Result of test* (R), *Test Decision* (T) and *Build Decision* (BD)) it can be expressed with a link restriction at the level of a link. This is a special case because DANs assume that a non existent variable does not have an impact on another variables. This means in our example that in the case the test result is not available it does not have any influence on *Build Decision*. In general DANs describe restrictions only at the level of a link, therefore they are not able to describe composed constraints taking into account previous decisions or observations.

Another difference between SIDs and DANs becomes evident from the restriction between the *Result of Test* and the *Build Decision*. DANs describe restrictions between the two variables which are involved in the restriction, therefore the model represents the cause-effect relationship of the restrictions. SIDs describe the constraint in the moment it becomes effective. In the case of a partial restriction, e.g. the restriction between the *Result of Test* and the *Build Decision* or when the variables which define the restriction do not appear consecutively the restriction can not be expressed with a simple label. In this case the second part of the guard is necessary to describe previous observations or decisions which condition the underlying arc. This can lead to large and complex descriptions and as guards are implicit, the model does not describe clearly between which variables the restriction appears. For example, the restriction between *Result of Test* and *Build Decision* is expressed at the outgoing arc of the *Build Decision*, while DANs describe the existence of the restriction on the link between the variables which define the restriction.

The reactor problem shows another special case of structural asymmetry. The constraint between the *Advanced reactor's components* and the *Result of Test* means that the test result can not be bad if the components of the reactor indicate success. DANs represent this constraint with a partial restriction, which describes that the *Result of Test* is restricted to a subset of its values, namely that the combination $\{as,b\}$ is not possible. In contrast SIDs describe this constraint assigning a zero probability value to this combination, because they can not describe constraints for variables which have a probability relation but are not observed at the decision scenario.

Conclusion

Both DANs and SIDs are able to represent order asymmetry efficiently as they allow parts of the model having a partial order of the decisions. We have seen that the use of clusters gives SIDs the ability to describe the temporal precedence for a group of decisions with unspecified temporal order clearer than DANs. This can make SIDs more suitable for the representation of problems such as the n-test problem. Nevertheless we have also seen that SIDs might have difficulties to represent problems where a chance variable is revealed by any of two decisions.

Regarding the description of structural asymmetry we have seen that both DANs and SIDs are able to describe structural constraints explicitly and avoid the use of dummy states. The approach that SIDs take to describe structural asymmetry is different from that of DANs. SIDs describe by means of the model of order under which conditions a variable appears, i.e., when a constraint becomes effective, while DANs describe the causes of the restriction, i.e., the incompatibility between two variables. Although the description of asymmetric constraints at the model of order is fully explicative, it is also more difficult to build such a model. When building the model of order the decision analyst must have in mind all the previous observations and decisions, that may cause a restriction on another variable. In contrast, the local description of restrictions focusing on the cause-effect relation between two variables is much more intuitive for human experts. In general the description of total restrictions on variables, such as the conditioned scenarios of the dating problem are easy to represent with a clustered decision tree and should not cause problems to the decision analyst. In contrast the representation of partial restrictions at a clustered decision tree is not intuitive.

In the representation of the reactor problem we have seen two types of asymmetry which are not straightforward to represent. We have seen that SIDs have problems for describing constraints for unobserved variables that have a probabilistic influence on other variables. SIDs solve this by assigning a zero probability value to the incompatible combination. In contrast DANs can describe explicit constraints for probabilistic relations using link restrictions. On the other hand, we have seen that partial restrictions or when the variables that define the restriction do not appear consecutively, the restriction cannot be expressed with simple labels above the arcs. In this case the second part of the guard is necessary to describe previous observations or decisions which condition the underlying arc, what can lead to large and complex descriptions. Additionally guards are shown when the restriction becomes effective and not where it is caused. For this reason the model does not show clearly which variables cause a restriction.

Another difference between DANs and SIDs is also that the model of order represents explicitly the information precedence. In contrast DANs describe implicitly and only up to a partial order the information flow by means of revelation arcs. The approach of DANs to substitute information arcs with revelation arcs is a local description of information precedence, which is easy to define for the decision analyst.

Regarding the use of labels to describe constraints and information flow, DANs do not show labels at the graphical level while SIDs use annotations above the structural arcs. DANs describe the occurrence of constraints or information flow depicting revelation arcs and link restrictions, but do not show the details about the values at the graphical level. For simple problems the description of constraints and information flow with labels at the graphical level is an advantage, as the model is more explanatory and easier to understand. For complex problems, where the variables have large state spaces the representation of labels is a drawback, as the diagram becomes complex and unreadable. In this circumstances the approach of DANs to hide the specification of constraints and information flow may result convenient .

In summary, we can conclude that SIDs are the main competitor of DANs as both formalisms provide efficient solutions for the representation of order asymmetry and structural asymmetry.

Their main difference lies in the presence of the model of order in SIDs and the way restrictions are described in this model. Although SIDs have the ability to use clusters to avoid the explicit description of irrelevant sequences of variables, the construction of the model of order is still a task for the decision analyst when building the model. DANs propose a local description for information precedence and restrictions and we consider that this approach can be an advantage of DANs for the representation of larger problems, where the construction of the model of order might require an extra effort from the decision analyst. We think that this can be an important difference between DANs and SIDs for building real-world applications. Nevertheless SIDs have an important advantage of DANs: they can represent temporal precedence easier by the use of clusters as we have seen at the comparison of the n-test problem. It would be interesting to analyze in more detail which kind of real-world problems result easier to represent with SIDs and DANs taking into account the above described characteristics.

3.4 Summary of the comparison

In the previous sections we have seen how DANs address the representation of asymmetric decision problems and how they compare favorably to the alternative formalisms. In this section we will summarize the results of the comparison. First we describe our conclusion about the features which make a formalism suitable for the representation of order and structural asymmetry. Next we give a statement of the more suitable formalisms and a short description of their strengths.

3.4.1 Representation of order asymmetry

Situations where the order of decisions and observations changes can not be represented with the common ID formalism. From the respective representations of the diabetes problem we have seen that the ability to represent order asymmetry depends on whether the formalism requires a total order of the decisions. As DANs, UIDs and SIDs do not require a total order, they can all represent order asymmetry. The solution is to represent the decision and the variable that this decision reveals directly in the model, what is a natural representation of the structure of a test problem. The formalisms that require a total order of decisions (IDs, extIDs, SVNs and AIDs) need to include artificial variables which represent all the possible orders of decisions and observations, what obscures the structure of the model and makes it impossible to represent large diagnosis problems, such as the n-test problem. AIDs are a special case, because they relax the requirement of a total order to a partial order only if the decisions are completely independent. In the case of the diabetes problem the test decisions are not independent and therefore the diabetes problem was represented with a total order.

The advantage of DANs and UIDs over the other formalisms is that they do not have information arcs. Therefore they are appropriate for representing decision problems such as the$n$-test problem, because they do not need to indicate which variables are known when making a decision. While UIDs require all parent decisions to be taken to know the state of a variable, DANs require only one decision to be taken to know the state of a variable. So DANs can represent problems where a variable is revealed by one of two decisions, what cannot be represented by UIDs.

3.4.2 Representation of structural asymmetry

The representation of asymmetric constraints has two objectives: conveying the asymmetric nature of the problem at the graphical level to the decision analyst and creating a consistent and efficient model of the problem that can be treated with computers. All formalisms which describe

constraints explicitly (SVNs, AIDs, SIDs and DANs) are in principle suitable for the representation of asymmetry as they have a compact representation of the states space of the variables and avoid dummy states. The way constraints are described is different for each formalism as we have explained in the respective representations of the dating and the reactor problems:

- IDs and UIDs do not describe constraints explicitly and follow the approach of an artificial symmetrization of the problem, what makes them unsuitable for the representation of structural asymmetry.
- extIDs combine the description of the uncertainty model of an ID with distribution trees which capture the asymmetric aspects by explicitly describing the conditioning scenarios.
- SVNs rely on a VN based model for the description of uncertainty and represent asymmetry with a compact graphical representation of the decision scenarios and indicator valuations for complex constraints.
- AIDs adapt the semantics of the arcs and nodes of IDs for the representation of non-sequential decisions and conditioned scenarios, adding labels to describe under which condition a variable appears.
- SIDs use an ID based model to describe uncertainty combined with a compact but explicit representation of the decision scenarios which make visible the information precedence and asymmetric constraints.
- DANs combine an ID based model for the description of uncertainty with an explicit representation of restrictions between variables and a local description of information precedence which both capture the asymmetric aspects of the problem.

The strength of a formalism depends further on more specific aspects such as the expressiveness of the description of constraints at the graphical level, the ability to describe different types of structural asymmetry, the scope and mechanism of the definition of constraints and the representation of information precedence.

Regarding the description of asymmetric constraints at the graphical level the use of labels for arcs and nodes describing conditions for its occurrence is very common, so for example AIDs, SVNs and SIDs use labeled links to describe constraints and information flow what makes these diagrams very expressive because the decision analyst can read directly from the diagram details about the asymmetric constraints of the problem. Nevertheless the labels are often composed and include the sequence of previous observations and decisions, so the description with labels can make the diagram difficult to read for larger problems. Although extIDs have a concise and detailed description of the asymmetric constraints at the functional level using distribution trees, they do not represent any information about the asymmetric constraints in the main diagram. Therefore they are not very useful as a communication tool because the decision analyst has to analyze several diagrams to get an understanding of the structure of the problem. DANs are also different from the other formalisms regarding the description of constraints at the graphical level. While the labels of the above mentioned formalisms (AIDs, SVNs and SIDs) describe conditions under which a variable occurs and are shown at the graphical model, DANs follow a different approach as they express the occurrence of constraints at the graphical level by focusing on the cause-effect relation of the restriction. Although DANs do not show the details of the specification of the constraints, they represent systematically all constraints and distinguish further whether the restriction is total (i.e., the variable is non-existent in a scenario) or if the restriction is partial (i.e., the variable appears with a subset of its values). In this aspect DANs overcome a

shortcoming of AIDs, which distinguish also these types of asymmetry (structural and functional in AID terminology) but represent only the first type systematically at the graphical level.

SVNs and SIDs take a different approach to describe asymmetry as they include a model for the description of the chronological order of observations and decisions. This model describes under which conditions a variable is observed, what is useful for the description of conditioned scenarios, i.e., when the variable may be non-existent in a scenario (i.e., totally restricted). In the case of a partial restriction or when the variables which define the restriction do no appear successively at the model of order the description with a clustered decision tree is not that straightforward as the occurrence of a variable with a subset of its values does not necessarily condition the appearance of another variable, what is basically the information represented in this model. SVNs use indicator valuations for this type of restrictions, which describes the incompatible combinations between the variables which define the restriction. DANs represent the restrictive influence of a variable also directly with link restrictions involving only the two concerned variables and using a more intuitive definition than that of indicator valuations. Although SIDs are able to describe this type of constraint concisely with composed guards, they have the drawback that the model does not reveal information about the restrictive influence between the involved variables because the guards are implicit, i.e., they are not shown between the variables which define the restriction.

Another difference we have seen during the comparison of the formalisms is that DANs use a different approach to describe information precedence. DANs describe locally information precedence. This is the opposite approach of SVNs and SIDs, which have an explicit description of the sequence of variables and AIDs, which use information arcs. This gives DANs an advantage when describing complex problems where a lot of variables might be known initially or revealed by a decision. An explicit representation of the information precedence for this kind of problems would create a complex and difficult to understand representation, while the DAN representation would remain simple.

Table 3.3 summarizes the main features of the different formalisms explained before. This table considers the explicit representation of the sequence of variables, the description of the independence relationships at the model, the use of dummy states and labeled links. It is also relevant whether the formalism requires a total order of the decisions and if it uses informational arcs.

	explicit sequence	causal relations	dummy states	labeled links	total order	inform. arcs
DT	yes	no	no	no	yes	yes
ID	no	yes	yes	no	yes	yes
extID	no	yes	yes	no	yes	yes
SVN	yes	depends	no	yes	yes	yes
AID	no	yes	no	yes	yes	yes
UID	no	yes	yes	no	no	no
SID	yes	yes	no	yes	no	yes
DAN	no	yes	no	yes	no	no

Table 3.3: Comparison of the main features of the different formalisms.

3.4.3 Advantages of some formalisms

In the comparison of the formalisms we have seen that only DANs and SIDs are able to represent both order and structural asymmetry, while UIDs only represent order asymmetry well, and AIDs and SVNs only address structural asymmetry. Therefore it depends on the characteristics of the decision problem, which formalism may be used. If order asymmetry is not an issue at the problem and only structural asymmetry appears AIDs, SVNs, SIDs and DANs provide all a good solution. Nevertheless only the DAN formalism is able to represent complex problems, what might be the explanation of why the other formalisms have not been used so far to represent real-world problems. In the following we briefly summarize the most important aspects of the formalisms we found most interesting (SVNs, AIDs, SIDs and DANs) and their main differences with DANs.

Strength of SVNs, AIDs and SIDs

SVNs have resulted suitable for the representation of structural asymmetry as indicator valuations are very powerful for describing complex constraints above all types of variables. Nevertheless the flexibility of indicator valuations is opposed to its usability. As indicator valuations number out all compatible combinations of states their specification is difficult and their reading is not intuitive. DANs at contrast specify link restrictions at the level of a link, what is independent of the complexity of the problem and intuitive for the decision analyst as he only has to consider the variables involved at a link. But SVNs have another important difference, i.e., they rely on VNs, which gives them flexibility to represent probabilistic models with any factorization, although they are also able to represent direct and conditional probabilities such as happens at the other ID based models. SVNs use further a compact decision-tree graph which shows the possible decision scenarios explicitly and therefore combine the representation of the probabilistic model with a model of order, what makes them very expressive.

AIDs have been shown to represent structural asymmetry, i.e., conditioned scenarios very clear, as they describe at the graphical level with labels the conditions under which an arc or node appears and with special arcs restrictions between variables. Although the meaning of restriction-arcs and test-arcs is similar to link restrictions and revelation arcs, the representation of restrictions and information flow of AIDs is based on labels and not all types of asymmetry are represented at the graphical level. In this aspect DANs express constraints better than AIDs as link restrictions can describe restrictions on all type of variables distinguishing further total and partial restrictions. Although the use of labels at the graphical level makes the model very expressive as the decision analyst can read directly the conditions for the appearance of a variable from the model, this information can make the diagram also unreadable when the problem is complex or the state space of the variables large. The representation of the conditions as labels however has been shown to be useful as AIDs can use this information to break the original problem into smaller problems, which describe the decision scenarios or the relevant variables for a single decision. Both the solution method and the decision analyst take profit from this decomposition, as the reading of the problem is easier for the decision analyst and the computations of the solution are less complex.

The comparison with SIDs has shown that they are suitable to represent both order and structural asymmetry. Order asymmetry is represented efficiently as SIDs allow parts of the problem having a partial temporal order. Constraints and information precedence are expressed with annotations above arcs, which describe explicit the set of scenarios and the conditions under which a variable occurs. SIDs are able to specify composed constraints and describe conditioned scenarios, but this approach does not describe the cause-effect relation of restrictions between variables. Therefore the description of asymmetric constraints with SIDs can be a difficult task

for the decision analyst as he needs to have in mind all previous observations and decisions when specifying the model of order.

Although SVNs, AIDs and SIDs have efficient approaches for the representation of structural asymmetry, they describe explicit information precedence. SVNs and SIDs use a compact but explicit description of the order of the variables and AIDs describe information precedence with information arcs which are effective in certain conditions. The explicit description of information precedence can make the diagram complex and unreadable when a lot of variables are known initially or revealed from a decision. Only the SID formalism proposes a solution for this problem, as this formalism has clusters to describe parts of the problem with an unspecified temporal order. We consider that the explicit description of the information precedence can be a difficulty for the decision analyst to construct the model for complex problems, what might explain why these formalisms were not used so far to represent real-world problems.

Strength of DANs

DANs overcome the limitation of the alternative formalisms regarding the description of information precedence as they represent locally information precedence with revelation arcs. We consider this feature of DANs a very important difference, which makes DANs a promising model for the representation of complex problems. The representation of order asymmetry takes also advantage of the use of revelation arcs. The combination of the flexibility to represent decisions with an undefined order and the local description of information precedence with revelation arcs make it possible to represent the decision and the variable the decision reveals directly at the model. This avoids the explicit description of all possible order of decisions and the correspondent relations of information precedence such as happens at other models, what gives DANs a large degree of flexibility to represent complex problems. For instance DANs are suitable to represent the n-test problem while most of the other formalisms have difficulties to represent this problem.

DANs have another important difference with regard to the description of structural asymmetry. At DANs the description of the restrictions and information flow centers on the conditioning relation between variables and not at the description of the scenario where a restriction becomes effective or an information state is used, mainly because DANs do not focus on the representation of order. The asymmetric aspects are described with link restrictions and revelation arcs at the level of a link what is independent from the complexity of the problem. These elements are intuitive to use as the decision analyst only has to think about which combinations of states between two variables are incompatible (link restrictions) or which values of a variable reveals the values of another variable (revelation conditions). This approach makes DANs easy to use and allows the construction of compact models which are built upon the essential relations between the variables of the domain.

Beside these main features which make DANs a suitable model we have seen a difference regarding the expressiveness of the graphical model. DANs have the approach to describe the existence of constraints and information flow at the graphical level by highlighting visually link restrictions and revelation arcs. Nevertheless the alternative formalisms which use labels at the graphical level are more expressive for simple problems as we have seen at the example of the dating problem. But the separation of the description of the occurrence from the details of their specification gives DANs the advantage of being independent from the complexity of the problem, because the appearance of too much labels can make a model unreadable.

3.5 Summary DANs

In this chapter we have presented the DAN formalism as a new probabilistic graphical model for the representation of asymmetric decision problems. We have introduced the definition of the formalism and the proposed evaluation method, which is currently the transformation into an equivalent decision tree. Next we have represented the three asymmetric decision problems introduced in the previous chapter to see how DANs address different types of asymmetry. We have also compared the solutions of the alternatives formalisms to the DAN solutions explaining the differences for the representation of order and structural asymmetry. This comparison has shown that DANs compare equally or even better with the other formalisms in all important aspects, being sequential influence diagrams (SIDs) their main competitors. DANs provide a natural representation of both order and structural asymmetry by focusing on the representation of the underlying relationships between variables instead of focusing on the sequential aspects of observations and decisions which are part of the solution. DANs therefore are a promising model for the representation of complex decision problems.

4 Implementation of DANs in OpenMarkov

DANs have been implemented at OpenMarkov, an open-source software tool for the edition and evaluation of PGMs. The choice to implement DANs at OpenMarkov was clear, as OpenMarkov is available to a wide audience as it is freely available and already known by the research community for the probabilistic models it already implements. This part of the work gives a brief introduction to OpenMarkov and describes the extensions that we have added for the implementation of DANs. We use UML to describe our work due to the object-oriented approach of OpenMarkov.

4.1 OpenMarkov

OpenMarkov is an open-source software tool for the edition and evaluation of PGMs developed by the Research Center on Intelligent Decision-Support Systems (CISIAD) of the UNED, in Madrid, Spain. OpenMarkov is based on the project Carmen (Arias & Díez, 2008; Arias, 2009) which started at 2003 and was renamed as OpenMarkov in 2010.

OpenMarkov supports several types of networks, such as Bayesian networks (BNs), Influence diagrams (IDs), Limited memory IDs (LIMIDs) and Markov networks. Currently OpenMarkov can only evaluate Bayesian networks, influence diagrams and Markov processes with atemporal decisions (MPADs). OpenMarkov implements also several types of temporal models, such as dynamic Bayesian networks, simple Markov Models (SMM), Markov Decision Processes (MDPs), Partially observable MDPs (POMDPs), Dec-PoMDP and dynamic LIMIDs. A detailed description of the representation of real-world problems with dynamical models can be found in (Díez & van Gerven, 2011) and (Díez et al., 2011).

OpenMarkov's default format for storing probabilistic models is ProbModelXML, which was proposed by Arias et al. (2011) as a common language for the specification of PGMs. ProbModelXML uses an XML syntax and allows to specify a large set of graphical models with a wide range of properties.

OpenMarkov has an object oriented design as it is programmed with Java and a modular organization as Maven is used for its construction. Figure 4.1 shows the organization of OpenMarkov in sub projects and the dependencies between the different projects:

- The project *core* implements the underlying data model for the representation of PGMs such as graphs and network types.
- The project *gui* implements the GUI of OpenMarkov.
- The project *io* contains the code for the writing and reading of networks in ProbModelXML format and Elviras format.
- The projects *learning.gui*, *learning.algorithm* and *learning.metric* are used for the automatic construction of models from data (databases).

- The project *inference* and *inference.heuristic* are used for the evaluation of a decision problem represented as PGM.

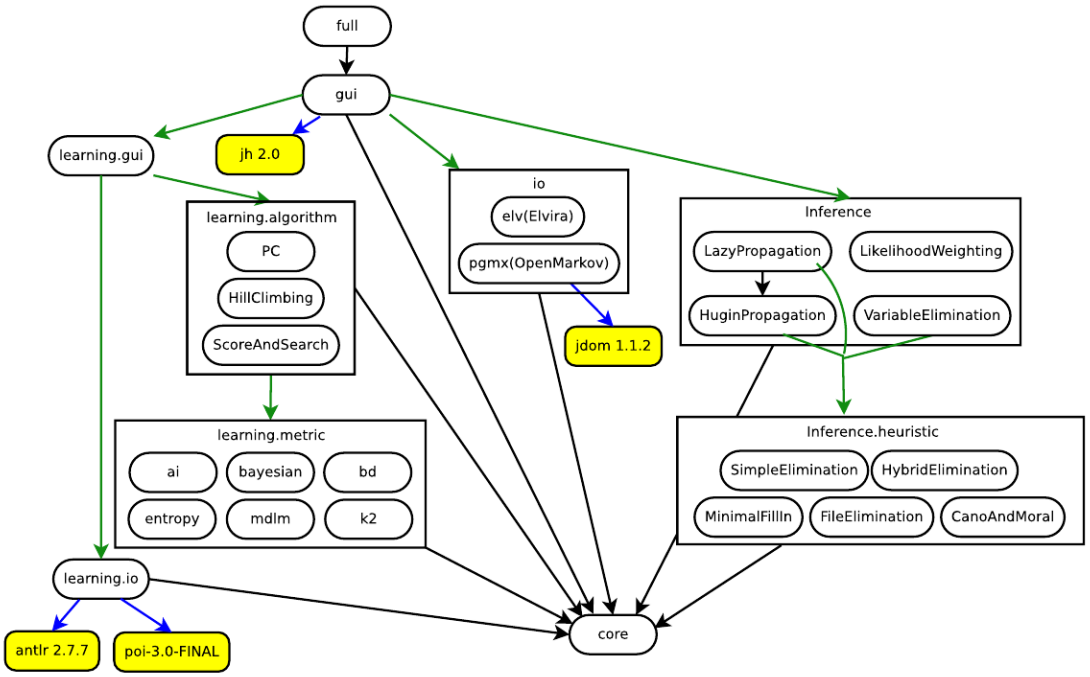


Figure 4.1: OpenMarkov organization: White rounded rectangles represent OpenMarkov subprojects. Black links indicate static dependencies between them. Blue links denote dependencies on external libraries, which are represented by yellow rounded rectangles. Green links denote dynamic dependencies discovered at runtime via Java annotations; for example, the GUI will detect dynamically the available inference algorithms, and exact inference algorithms will detect automatically the available elimination heuristics. This way it is possible to extend OpenMarkov by adding subprojects (artifacts, in maven's terminology) that will behave as plug-ins.

4.2 Extensions to implement DANs

This section describes the extension of OpenMarkov to implement DANs. The extensions are centered mainly on the establishment of a new network type and the additional features of DANs with regard to IDs, which are in particular link restrictions, always observed variables and revelation arcs. The following list describes briefly the new requirements of the software system with regard to DANs. The system should provide the following possibilities:

1. Create DANs.
2. Store the configuration of a DAN in the ProbModelXML format, including link restrictions, always-observed variables and revelation conditions.
3. Read the configuration of a DAN from a ProbModelXML file and represent the graphical model.

4. Represent graphically the model of a DAN:
   - Links having a link restriction are depicted with a single/double crossing stripe.
   - Always observed variables are depicted by painting the border with a special color. The system uses this same color for painting the always-observed variables and revelation arcs. In the future, this color might be configurable by the user.
   - Links having revelation conditions are painted with a special color.
5. Edit link restrictions:
   - The edition should be graphically and intuitive and the system must store them in an appropriate data structure.
   - The existence of link restrictions has an impact on the conditioned probability distributions of the second node of the link. The conditional probability table should highlight the states, which are impossible in some scenarios according to the restrictions imposed by a link restriction. Impossible states have the probability value zero and appear at not-editable cells, which are highlighted in a special color.
6. Configure the always observed property for a variable:
   - The edition should be graphically, i.e. at the node properties panel. The system should store these variables in an appropriate data structure.
7. Edit revelation conditions:
   - The GUI should allow to edit the revelation conditions graphical in an intuitive way. The edition for finite state and discretized variables consists of the selection of the states and the edition for numeric variables compromises the edition of intervals. The system should store revelation conditions in an appropriate data structure.

Due to the object-oriented approach of the project, we use UML diagrams for the description and modeling of the software system regarding the implementation of DANs.

4.3 Specification of the software system

Table 4.1 shows a summary of the use cases, which describe the system behavior related to DANs and which are explained in detail in the correspondent sections.

Use case Id	Use case name	Scope	Description
1	Create DAN	system-wide	Creates a new instance of DAN
2	Store DAN	system-wide	Write the configuration of a DAN in a ProbModelXML file
3	Read DAN	system-wide	Read the configuration of a DAN from a ProbModelXML file
4	Represent graphical model	system-wide	Paint graphical model of the DAN
5	Paint link restriction	link	Depict link restriction
6	Paint revelation arc	link	Depict revelation arc
7	Paint always observed variable	variable	Depict always observed variable
8	Edit link restrictions	link	Set compatibility values to a combination of variables
9	Remove link restriction	link	Removes the link restriction of a link.
10	Represent incompatible states at the conditional probability table (CPT)	prob. table	Adapt the representation of CPT to link restrictions
11	Configure Always Observed prop.	variable	Edition of the always observed property
12	Define Revealing conditions	link	Defines the revelation conditions for a link
13	Define Revealing states	link	Edition of revealing states
14	Define Revealing intervals	link	Definition of revealing intervals

Table 4.1: Use case overview.

Figure 4.2 shows the use case diagram for the new feature of the software system. The detailed description of the use cases shown in Table 4.1 can be found in the appendix A.2.

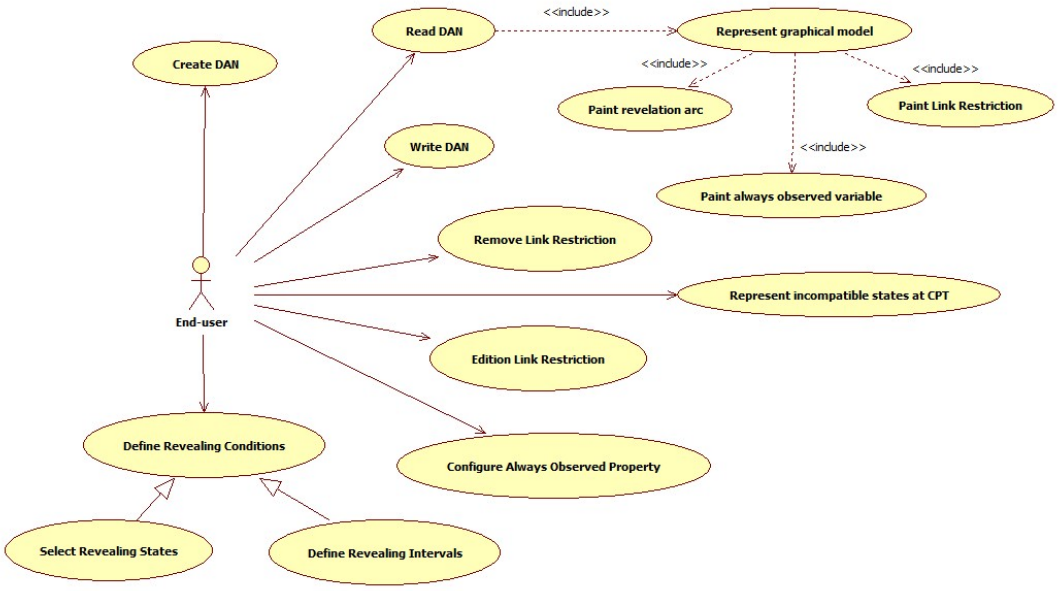


Figure 4.2: Use cases diagram.

4.4 Analysis and design

This section focuses on the analysis and design for the requirements specified in the previous sections. The implementation of the new features require modifications of the data model, the GUI and the encoding of the graphical model in ProbModelXML format. Table 4.2 shows which subprojects of OpenMarkov are modified in each case.

Modifications	sub project
Data model	core
GUI	gui
ProbModelXML	io

Table 4.2: Modification of source code.

4.4.1 DAN network type

In OpenMarkov, the properties of a network type are defined by means of constraints. The configuration of constraints for a network type is important as the access to functionalities at OpenMarkov is based on constraints and not on the network type. This modular approach based on constraints gives OpenMarkov the capability to implement new network types with ease. The Table 4.3, taken from Arias et al. (2011), shows the configuration of the constraints for the different network types. This Table shows that DANs are similar to IDs, as they share almost all the same constraints, but DANs have the additional capability of restrictions on links, revelation arcs and always observed variables. Therefore the default value for the constraints NoRestriction and NoRevelationArc is overridden for the Decision Analysis Network type so that DANs do not have these constraints.

	BayesianNetwork	MarkovNetwork	InfluenceDiagram	LIMID	**DecisionAnalysisNetwork**	DynamicBayesianNetwork	MPAD	MDP	POMDP	Dec-POMDP	DynamicLIMID
NoEmptyName	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DistinctVariableNames	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
OnlyFiniteStatesVariables	O	O	O	O	O	O	O	O	O	O	O
OnlyNumericVariables	O	O	O	O	O	O	O	O	O	O	O
OnlyChanceNodes	**Y**	**Y**	N	N	N	**Y**	N	N	N	N	N
OnlyOneUtilityNode	–	–	O	O	O	–	O	O	O	O	O
OnlyAtemporalVariables	Y	Y	Y	Y	Y	**N**	**N**	**N**	**N**	**N**	**N**
OnlyTemporalVariables	N	N	N	N	N	**Y**	N	**Y**	**Y**	**Y**	**Y**
OnlyOneAgent	Y	Y	Y	Y	Y	Y	Y	Y	Y	**N**	Y
DistinctLinks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NoMultipleLinks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
OnlyDirectedLinks	Y	**N**	Y	Y	Y	Y	Y	Y	Y	Y	Y
OnlyUndirectedLinks	N	**Y**	N	N	N	N	N	N	N	N	N
NoRestriction	Y	Y	Y	Y	**N**	Y	Y	Y	Y	Y	Y
NoRevelationArc	Y	Y	Y	Y	**N**	Y	Y	Y	Y	Y	Y
NoSelfLoop	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NoCycle	Y	**O**	Y	Y	Y	Y	Y	Y	Y	Y	Y
NoClosedPath	O	O	O	O	O	O	O	O	O	O	O
MaxNumParents	O	O	O	O	O	O	O	O	O	O	O
NoUtilityParent	–	–	O	O	O	–	O	O	O	O	O
NoSupervalueNode	–	–	O	O	O	–	O	O	O	O	O
NoMixedParents	–	–	O	O	O	–	O	O	O	O	O
NoBackwardLink	–	–	–	–	–	Y	Y	Y	Y	Y	Y

Table 4.3: Constraints used in the OpenMarkov tool. The letter in each cell indicates whether a constraint is associated with a network type: Y = yes, N = no, O = optionally. A dash means that a constraint does not make sense because of the presence of another constraint (see the text for a more detailed explanation). Each constraint has a default behavior; a bold letter in this table means that the default behavior has been overriden for a particular type of network.

Arias (2009) gives a detailed description of the architecture of the constraint framework of OpenMarkov (formerly Carmen), which is built upon constraints relative to the structure of the graph or regarding nodes and variables. These constraints are controlled during the edition of the probabilistic network to assure the correctness of the probabilistic network. Figure 4.3 shows the main classes of the constraint framework, where the entity *ProbNet* is associated with several *PNConstraints*, each responsible of the control of a certain constraint. This section provides the specification of several new constraints proposed by Arias et al. (2010) (described as realizations of PNConstraint in Figure 4.3), which were implemented in the context of this project in order to make the constraint framework operative:

Constraints about links (structure of the graph)

- MaxNumParents: It limits the number of parents that a node may have, as specified by its argument.

- NoClosedPath: It forbids both cycles and loops. In the case of undirected graphs, this constraint implies that the graph of the model is a tree. In the case of directed graphs, it implies that the graph is polytree.

- NoRevelationArc: Revelation arcs are used by DANs to indicate that a variable reveals the value of another variable ; for example, the decision of performing a test "reveals" the result of the test. A network that admits revelation arcs can also indicate that some variables are always observed . Currently DANs are the only network type that admits revelation arcs and always-observed variables.

- NoRestriction: A constraint (x, y) associated with an arc $X \rightarrow Y$ means that the values x and y are incompatible ; for example, the decision of discharging a patient now (x) prevents the performance of a test a few hours later (y). Currently all the network types have this constraint, except decision analysis networks.

- NoMixedParents: This restriction establishes that all the parents of a utility node belong to only one of these two sets of parents: (1) chance and decision nodes, or (2) utility nodes.

- NoMultipleLinks: This constraint makes the undirected link $A - B$ incompatible with both $A \rightarrow B$ and $B \rightarrow A$; however $A \rightarrow B$ would be compatible with $B \rightarrow A$. This restriction will be used, for instance, when building chain graphs.

- NoUtilityParent: This constraint forbids that a chance or decision node has a utility node as a parent. However, it does not forbid that a utility node be a parent of another utility node (the latter would be called *supervalue* node Tatman & Shachter (1990)).

- DistinctLinks: The network cannot have two equal links. For example, the network cannot have two links $X \rightarrow Y$. A link $X \rightarrow Y$ is different from $Y \rightarrow X$ and $X - Y$, but the latter is considered to be the same as $Y - X$.

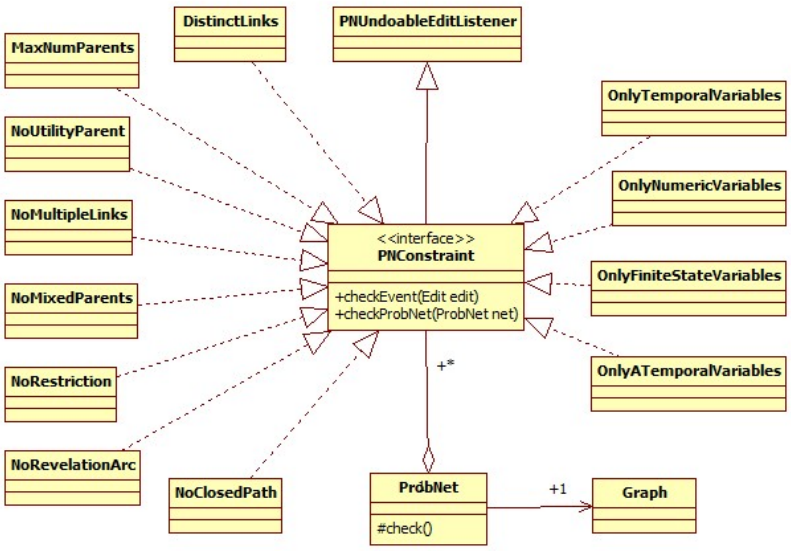


Figure 4.3: Class diagram constraints framework

Constraints about nodes and variables

- OnlyATemporalVariables: The model contains no temporal variable.
- OnlyFiniteStateVariables: All the chance and decision variables must be of finite-state or discretized. However, this restriction does not affect utility variables, which in ProbModelXML are assumed to be always numerical.
- OnlyNumericVariables: All variables must be purely numeric. Discretized variables are not admitted, because they are treated as finite-state variables.
- OnlyTemporalVariables: Used by most dynamic models. In the current version of ProbModelXML , the only type of dynamic model that does not have this constraint is SimpleMarkovModel , which accept both temporal and atemporal variables.

4.4.2 Link restrictions

This section explains the modifications on the data model, GUI and the ProbModelXML relative to link restrictions.

Definition and preconditions

A restriction expresses the incompatibility between certain values of two variables connected by a directed link $A \rightarrow B$. The restrictions associated to that link are represented by means of a potential ψ defined on $A \times B$, such that $\psi(a, b) = 1$ expresses compatibility and $\psi(a, b) = 0$ incompatibility. A link restriction can only exist if it satisfies the following conditions:

- Variable A must be chance or decision.

- The variables A and B must be finite-states. In this master thesis we will deal only with restrictions for finite-states variables (including discretized variables, which are treated as if they were finite-states).
- The network does not have the NoRestriction constraint. The NoRestriction constraint prevents a network from having restrictions. Currently all the network types have this constraint, except decision analysis networks, what means that only DANs can have restrictions.

Modifications of the data model

The modifications of the data model comprise the data structure to store link restrictions and the related methods for retrieving and assigning the information. Figure 4.4 shows the class diagram for the class Link, the entity which is the container of the link restrictions.

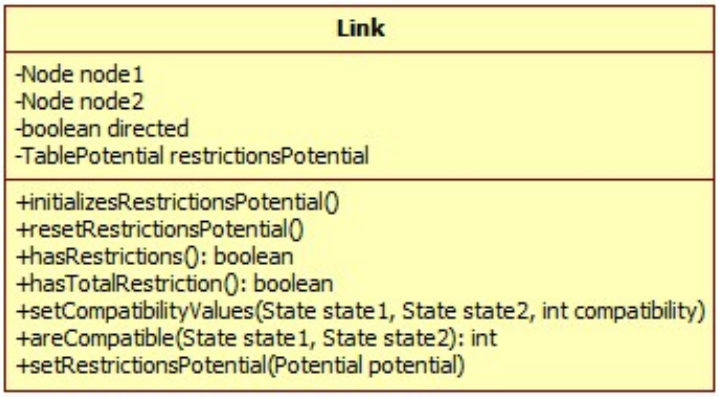


Figure 4.4: Adaption of the class Link for storing restrictions.

The class Link has an attribute named restrictionsPotential, which is a potential ψ defined on its variables, Var1 and Var2, for storing the compatibility values. This potential stores the value 1 for compatible pairs of values and 0 for incompatible values. The corresponding attribute has a null value when the link has no associated restrictions. The class Link has several methods for assigning and retrieving information about the restrictions:

- hasRestrictions(): boolean This method indicates if the link has a restrictionsPotential. The method checks whether the potential has a null value or not.
- hasTotalRestriction(): boolean This method indicates if the link has a restriction Potential and if the restriction is total.
- initializeRestrictionsPotential(): This method associates a Potential to the link. If Var1 and Var2 are finite-states variables, this method initializes the attribute restrictionsPotential with a TablePotential whose variables are Var1 and Var2 and whose values are all 1 (compatible).
- resetRestrictionPotential(): This method assigns a null value to the restrictionsPotential if all its values are compatible.

- setCompatibilityValues(State state1, State state2, int compatibility): This method assigns the value of the parameter compatibility to the combination of the variables Var1 and Var2. The compatibility value should be 1 for compatibility and 0 for a restriction.
- areCompatible(State state1, State state2): boolean This method returns the compatibility value of the combination of the variables Var1 and Var2. It returns 1 if the combination is compatible and 0 otherwise.
- setRestrictionsPotential(Potential potential): This method assigns a potential to the attribute restrictionsPotential.

Modifications of the GUI

The modifications of the GUI for the representation of link restrictions comprise the following functionalities, which are described subsequently with more detail:

- Configure the contextual menu options of a link (Use case 8 and 9).
- Paint link restrictions (Use case 5).
- Edition of link restrictions (Use case 8).
- Representation of impossible states at the CPT (Use case 10).

Contextual menu options The activation of the menu options is a responsibility of the class EditorPanel (package org.OpenMarkov.core.gui.window.edition) as described in the sequence diagram in Figure 4.5.

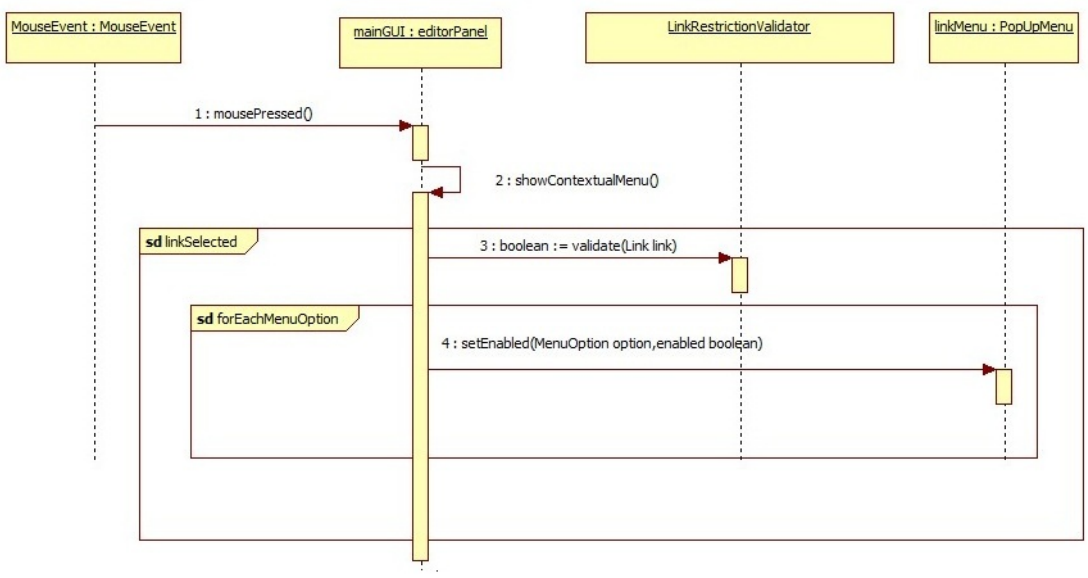


Figure 4.5: Diagram for the activation of the link restriction menu options.

This diagram shows that the process is initialized when a mouseEvent is received. The method showContextualMenu() determines if the mouse selection corresponds to the selection of a link.

If a link is selected the method isValid() of the class LinkRestrictionValidator is responsible for checking whether the link can have link restrictions. This class considers the preconditions of link restrictions explained at the definition of the link restriction (see Section 4.4.2). Depending on whether the link has already a link restriction, the different contextual menu options are activated. The option *Edit restrictions* and *Remove restrictions* are activated if the link has already a link restriction: otherwise only the option *Add link restrictions* is activated.

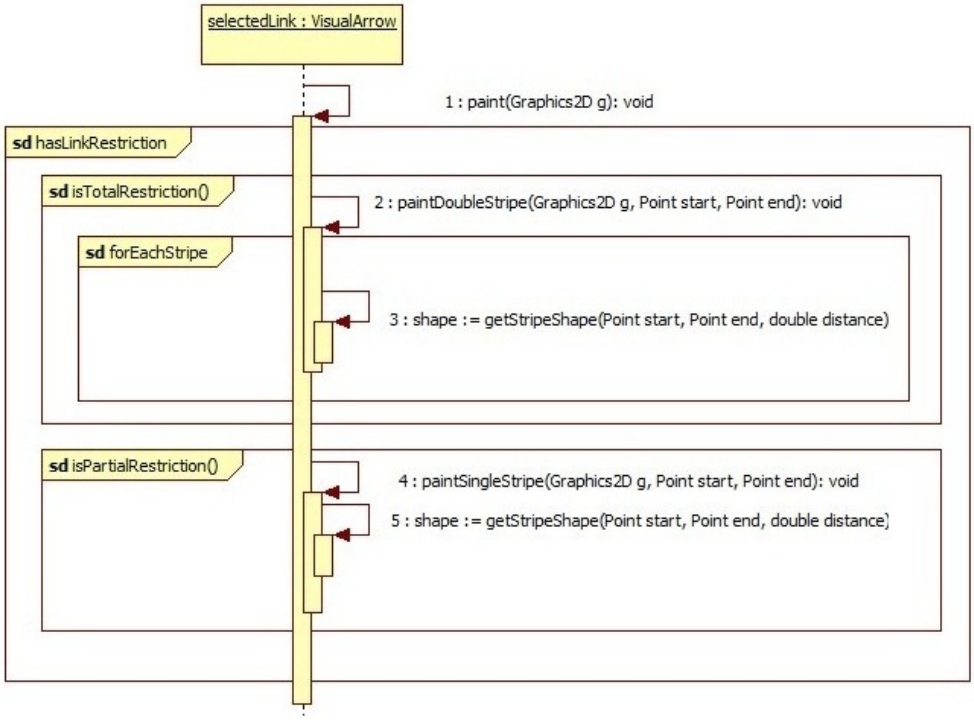


Figure 4.6: Diagram for the painting of a link restriction.

Paint link restrictions The GUI of OpenMarkov uses the shape drawing features from the Java 2D API. The nodes and links are represented basically using geometric primitives such as point, lines and shapes, which are painted in a Graphics2D instance. The representation of a link restrictions with stripes crossing the link comprises the painting of lines centered at the middle of the link each one at a given distance and crossing the link perpendicular. The dynamical model for the painting of the link restriction is described in Figure 4.6. The method paintDoubleStripe() or paintSingleStripe() of the class VisualArrow are responsible for painting the stripes and use the auxiliary routine getStripeShape() to get the shape object of the stripes. This method calculates the coordinates for a stripe located at a given distance d from the middle of the start and end point and with a perpendicular orientation. Graphics2D uses affinity transformations to calculate transformed coordinates, which include the following geometric transformations to calculate the transformed coordinates of a stripe.

- Translation to the origin of coordinates from the middle of the two points.
- Rotation of αcalculated as the gradient of the line according to $tan(\alpha) = \frac{\Delta(y)}{\Delta(x)}$.
- Translation to the distance d at the$x$-axis from the origin.

Edition of link restrictions The edition of the compatibility values of a link restriction opens a dialog which shows a table with the compatibility values (CVT) for the combination of states of the link variables. The implementation of this functionality should be based on the existing code for the representation of the conditioned probability tables (CPT). The reason for the reuse of the code is that both functionalities represent the combination of the states of variables in a table and that the values to represent are stored in a potential. Therefore the classes used to represent the compatibility values table are extensions of the existing classes which adapt certain features to the new requirements. In particular the differences are that the cells of the compatibility values are painted with green or red color according to its value and that the cells are not editable. The cells are sensitive to mouse events and on receiving a double click event change the value of the cell. Further the CPTs use a list of potentials to represent the conditioned probability, because a node can have several parents. In contrast, compatibility values exist at a level of a link and therefore only one potential is involved. The extensions of the existing code is described in Figure 4.7, where the relevant methods of each class are depicted.

The classes shown in Figure 4.7 override functionalities of their parent classes:

- LinkRestrictionCellRenderer overrides the setCellColors() method so that the cells are painted with red and green color according to its value.

- LinkRestrictionPanel adapts the method setData() as each CVT corresponds to only one potential.

- LinkRestrictionsValueTable overrides the method setValueAt() as the events of modifications must be handled with the specific object LinkRestrictionPotentialEdit. The methods inherited from PNUndoableEditListener are also overridden.

- The TableModel class (LinkRestrictionValuesTableModel) overrides the methods isCellEditable() and getColumnClass() so that cells are not editable and the CVT shows integer values.

Figure 4.8 shows the dynamic model for the edition of a compatibility value. The sequence is started by an mouse event received at the table. The table calls the setValueAt() method, which creates an SimplePNEdit object (LinkRestrictionPotentialEdit) to store the modified data at the link. The SimplePNedit object is handled by means of the PNESupport mechanism, so that that the modifications can be undone and redone. The PNESupport class calls the doEdit() method of the SimplePNEdit object which stores the modified compatibility value at the link.

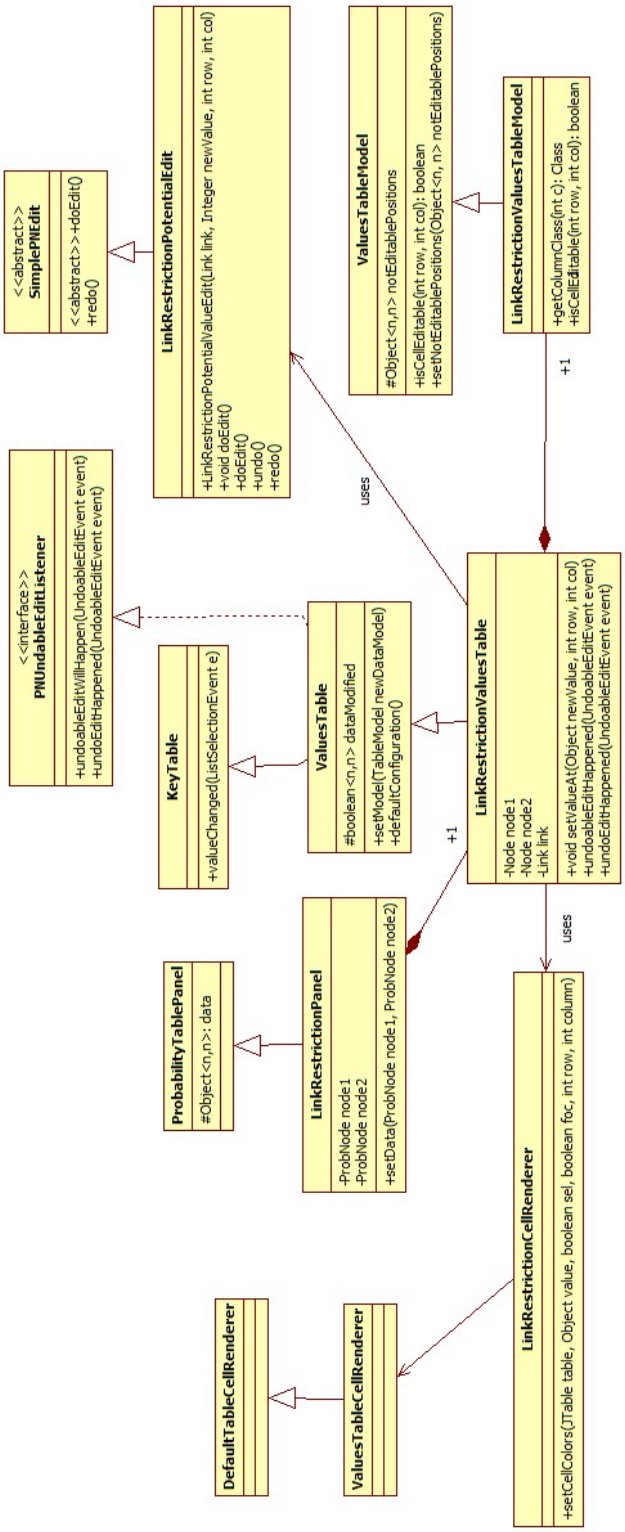


Figure 4.7: Class diagram for the GUI elements of the CVT.

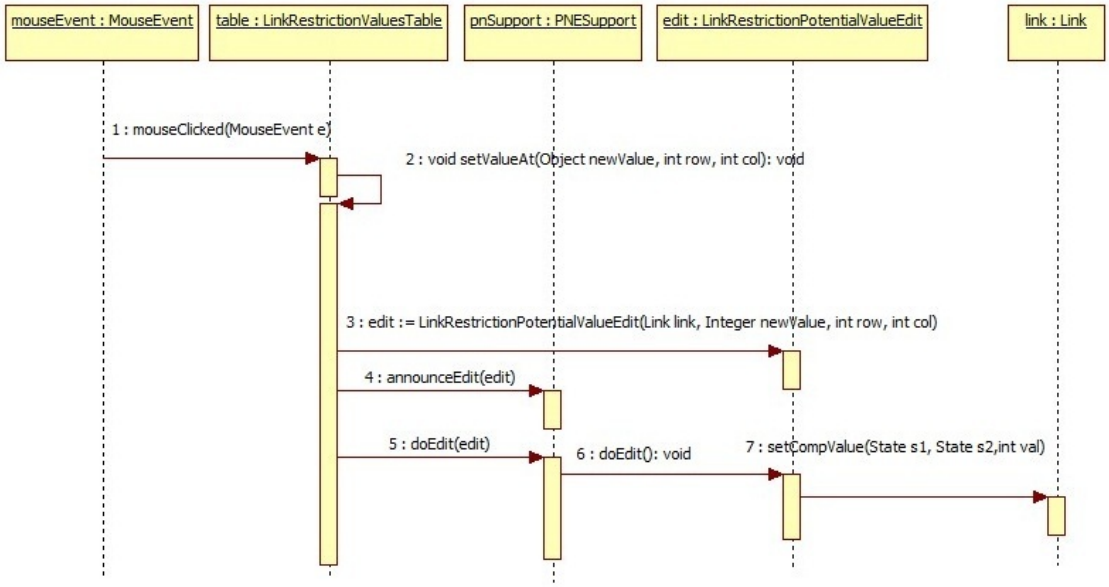


Figure 4.8: Diagram for the edition of a compatibility value.

Representation of impossible states at the CPT Figure 4.9 shows the class diagram of the entities involved in the representation of the conditioned probability table (CPT). For each class it describes the methods or attributes modified to adapt the representation of the CPT table to indicate impossible states. The control of the graphical representation of the impossible states is a responsibility of the TableModel, in particular of the class ValuesTableModel, which uses a data structure with information about non-editable positions to remark the impossible states.

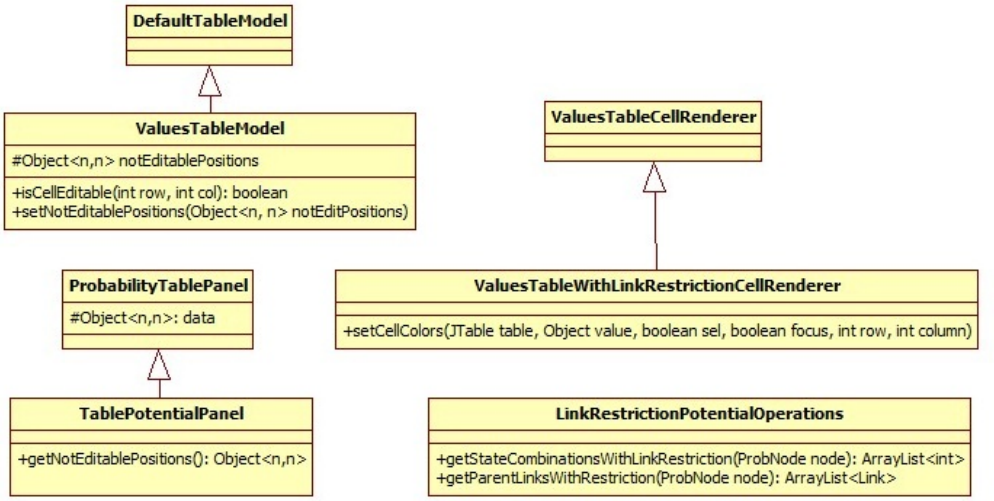


Figure 4.9: Class diagram for the GUI elements of CPT table.

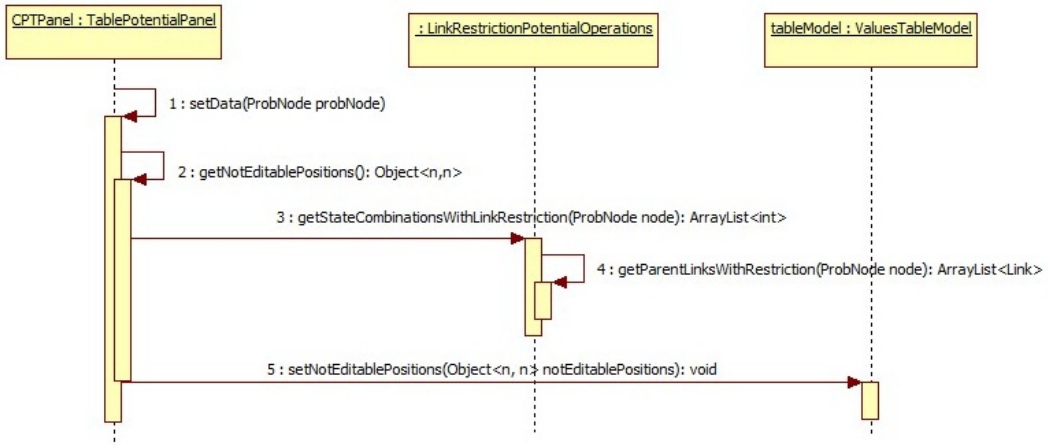


Figure 4.10: Diagram for showing impossible states at the CPT.

Figure 4.10 shows the sequence of operations during the initialization of the TableModel object. TablePotentialPanel initializes the TableModel with the content of the probability potentials in the method setData(). In order to represent additionally the impossible states because of the link restrictions an additional data structure with this information is provided from TablePotentialPanel to the TableModel. The class TablePotentialPanel uses the method getNotEditablePositions() to create a data structure with information about not editable positions at the TableModel due to link restrictions. The computations regarding the impossible states is responsibility of the auxiliary class LinkRestrictionsOperations and TablePotentialPanel converts this information to a table model readable format and provides it to the TableModel. The class ValuesTableModel uses this information about the not editable positions at its method isCellEditable(). This information is further used at the ValuesTableCellRenderer class which depicts non-editable cells with a special color.

```
<Link directed="true ">
    <Variable name="X"/ >
    <Variable name="Y"/ >
    <Potential type="Table" role="Restrictions" >
        <Variables >
            <Variable name="X">
            <Variable name="Y">
        < /Variables>
        <Values>1 1 1 0</Values>
    </Potential>
< /Link>
```

Figure 4.11: ProbModelXML code for a link restriction.

4.4.3 Modifications of ProbModelXML

ProbModelXML encodes link restrictions as an attribute of the link element. The link element contains a Potential with the role Restrictions to hold the link restriction's potential. Figure

4.11 shows the XML structure of a link with a restriction potential. The modifications of ProbModelXML concern the classes PGMXReader and PGMXWriter, which should be adapted to implement the reading and writing of a potential associated with a link in the format shown at figure 4.11. The writing of the potential must check if the values of the restrictionsPotential are all 1 and then this potential should not be stored with the link.

4.4.4 Always-observed variables

Definition and preconditions

A chance variable can be always-observed, which means that its value is known without taking any action. Always-observed variables have the following preconditions:

- Always-observed variables can only exist when the network does not have the constraint NoRevelationArc.
- The variable must be a chance variable.

Modifications of the data model

The property alwaysObserved is a new boolean attribute of the class Variable, which implements the correspondent setter and getter method (see Figure 4.12).

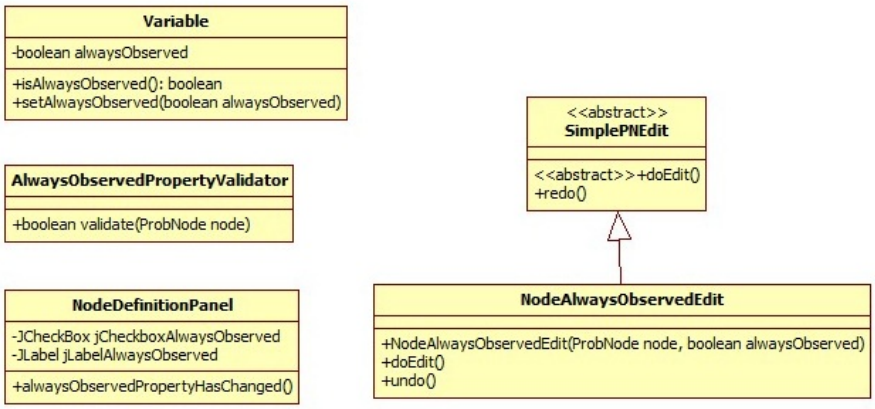


Figure 4.12: Class diagram for NodeDefinitionPanel.

Modifications of the GUI

The modifications of the GUI comprise the adaption of the node properties panel to show the always-observed property (Use case 11) and the modifications of the painting of the node (Use case 7).

Modifications of the node properties panel Figure 4.12 shows the entities involved in the graphical representation of the always-observed property. The class NodeDefinitionPanel is the GUI element which shows the label and the checkbox to configure the always-observed property. NodeDefinitionPanel uses the class AlwaysObservedPropertyValidator to validate if a node can have the always-observed property and activate the correspondent GUI elements.

In particular its method validate() checks if a variable satisfies the preconditions of an always-observed variable. The modifications of this property from the GUI is carried out by the class NodeAlwaysObservedEdit, which implements the SimplePNEdit interface.

Figure 4.13 shows the sequence of operations for the edition of the always-observed property. The sequence is started by an action event received at the NodeDefinitonPanel. The panel checks whether the actionEvent comes from the checkbox related to the always observed property and calls the alwaysObservedPropertyHasChanged() method, which creates a SimplePNEdit object (NodeAlwaysObservedEdit) to store the modified property at the variables. The SimplePNEdit object is given to PNESupport, which calls the doEdit() method of the SimplePN object, which stores the value of the always observed property at the variable. The use of a SimplePNEdit object assures that the redo() and undo() functions of the GUI work properly, as the modifications are handled with the PNESupport mechanism.

Paint always observed node The always-observed property is represented visually by drawing the node with a broader border and a special color. The class VisualChanceNode (package org.OpenMarkov.core.gui.graphic) needs to adapt its paint routine to take into account these requirements when painting the node.

Modifications of ProbModelXML

ProbModelXML encodes the always-observed property as an attribute of the variable element. Figure 4.14 shows the XML structure of an always-observed variable which should be implemented at the PGMXReader and PGMXWriter class.

<Variable name="Symptom" ...>
<AlwaysObserved>

Figure 4.14: ProbModelXML code for an always-observed variable.

4.4.5 Revelation arcs

Description and preconditions

A revelation arc between the node A and B is a link property and denotes that certain values of A reveal the values of B. These values are termed *revealing conditions* and depending on the type of the variable they are states or numeric intervals. In the case of discretized or finite state variables the revelation conditions reference states of the variable A (*revealing states*), which reveal the value of the variable B. In the case of numeric variables the revelation conditions are numeric intervals, i.e. closed, open or half-bounded intervals. The variable A reveals the value of the variable B when the value A takes on falls into any of these intervals (referred to as *revealing intervals*).

Graphically the revelation arcs are painted with the same color used to draw the border of the nodes that represent always-observed variables. A revelation arc mus fulfill the following preconditions:

- A revelation arc may only exist when the network does not have the constraint NoRevelationArc.
- A revelation arc may only exist when the first node is a chance or decision node and the second is a chance node.

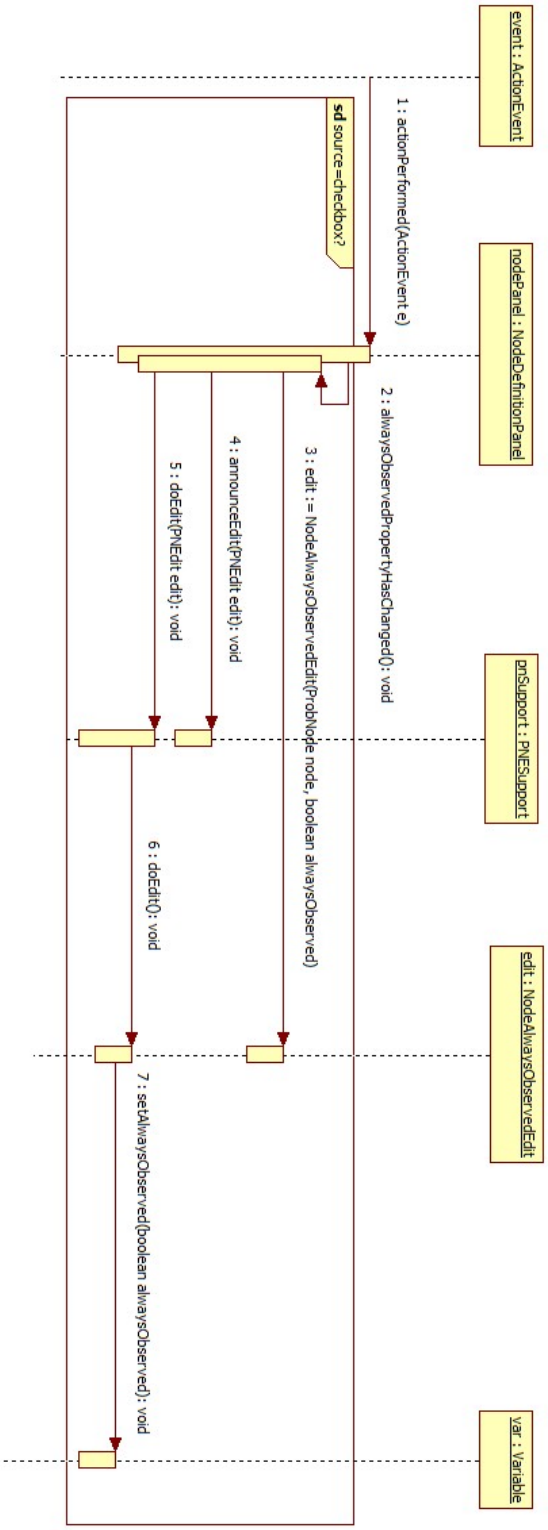


Figure 4.13: Diagram for the modification of the always-observed property.

Modifications of the data model

When a link between the nodes A and B is a revelation arc it is necessary to store the list of revealing values of the node A. Depending on the variable type the class Link stores the revealing values as objects of the type State or Interval. Figure 4.15 shows the class diagram of Link with the attributes and methods related to the revelation conditions.

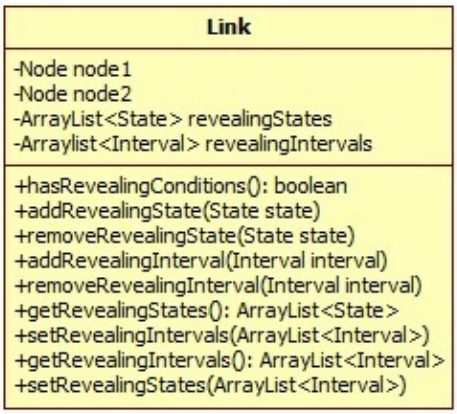


Figure 4.15: Adaption of the class Link to include revelation conditions.

The class Link contains the following methods to associate the revealing conditions to the link and to assign and read the revealing values:

- hasRevealingConditions(): boolean This method indicates whether there are revealing conditions for the link. The method returns true if the list of revealing states or revealing intervals is not empty.

- addRevealingState(State) and removeRevealingState(State) These methods add and remove a state from the list of revealing states. These methods should be used when the node is a finite state or discretized node.

- addRevealingInterval(Interval) and removeRevealingInterval(Interval) These methods add and remove an interval from the list of revealing intervals. These methods should be used when the node represents a numeric variable.

- getRevealingStates(): ArrayList<States> and setRevealingStates(ArrayList<States>). These methods are the getter and setter method for the list of revealing states.

- getRevealingIntervals(): ArrayList <Interval> and setRevealingIntervals (ArrayList <Interval>). These methods are the getter and setter method for the list of revealing intervals.

Modification of the GUI

The modifications of the GUI comprise the definition of the revealing conditions (Use case 12), which can be depending on the domain of the variable either the definition of revealing states (Use case 13) or the definition of revealing intervals (Use case 14) .

Definition of revealing conditions The edition of the revelation conditions of a link is accessible from a contextual menu option of a link. The option *Edit revelation condition* is available when the link satisfies the preconditions of a revelation arc. This validation is implemented at the class RevelationArcValidator at the method validate (See class diagram of figure 4.16). The class EditorPane, which represents the graphical pane, uses this method to activate the contextual menu option of a link.

The class RevelationArcPanel is the graphical window, which contains the GUI elements for configuring the revealing conditions. In the case the variable has a finite state or discretized domain, this panel uses a SelectableKeyTablePanel to represent graphically the interface for the selection of the states of the variables. If the variable has a numeric domain the RevelationArcPanel shows a RevelationArcDiscretizeTablePanel, which allows to configure the intervals.

The GUI element responsible for the configuration of the revealing states (SelectableKeyTablePanel) is an extension of the PrefixedKeyTablePanel, which is used at the domain definition of a finite state variable. This class is more specific as it uses a SelectableTableModel, so that the user can select different states by means of a checkbox . The modifications of the revealing states of a link are carried out at the class RevelationStateEdit, which implements the interface SimplePNEdit. The GUI element responsible for the configuration of the revealing intervals (RevelationArcDiscretizeTablePanel) is an extension of the the DiscretizeTablePanel, which is used at the definition of domains for numeric variables. This class adapts the edition of the intervals defined by the super class, so that the modifications are stored by means of a RevelationIntervalEdit object at the link. The class RevelationArcDiscretizeTablePanel implements several methods which correspond to the actions of the *Add* and *Remove* button and the edition of the intervals of the table.

Definition of revealing states Figure 4.17 shows the sequence of operations for the edition of the revealing states of a variable (see Section 4.4.5 for the definition). The sequence is started when a TableModelEvent is detected. The SelectableKeyTablePanel creates a SimplePNEdit object (RevelationStateEdit) initialized with the values according to the user selection on the table. This SimplePNEdit object is given to the PNESupport, which starts the edition encapsulated at the SimplePNEdit object. The modification of the state (Add or Remove) is carried out assigning the new value to the link.

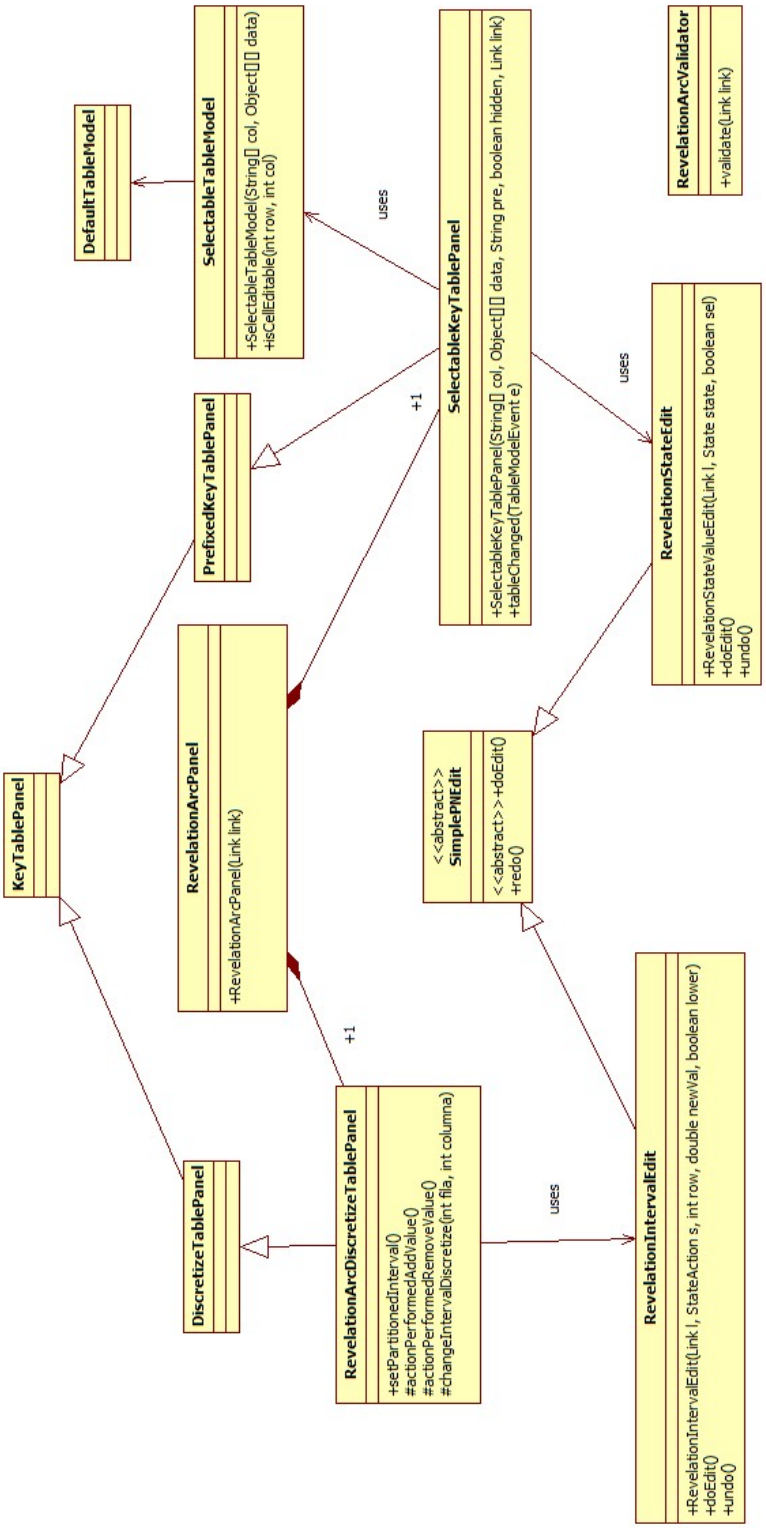


Figure 4.16: Class diagram for the edition of revelation conditions.

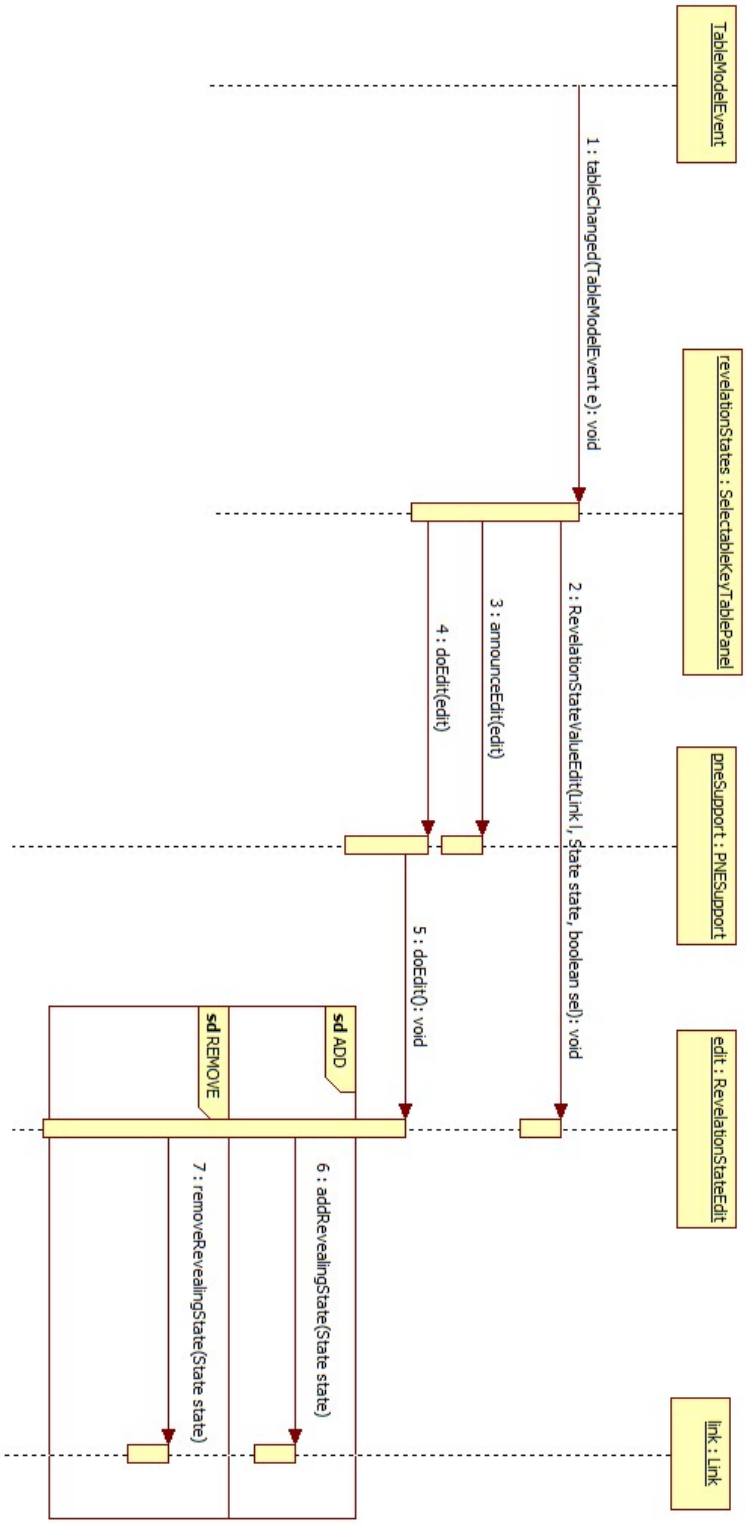


Figure 4.17: Diagram for the edition of revelation states.

Specification revealing intervals Figure 4.18 shows the sequence of operations for the edition of the revealing intervals (see Section 4.4.5 for the definition). The class RevelationArcDiscretizeTablePanel initializes the modifications when it receives an event from the different GUI elements of the window (buttons or table) and creates a SimplePNEdit object (RevelationIntervalEdit), which encapsulates the values of a interval according to the modification of the user. The SimplePNEdit object is given to the class PNESupport, which starts its edition by calling the doEdit method of the SimplePNEdit object. The modifications of the revealing intervals are stored at the correspondent link according to the type of modification (Add, Remove or Modify).

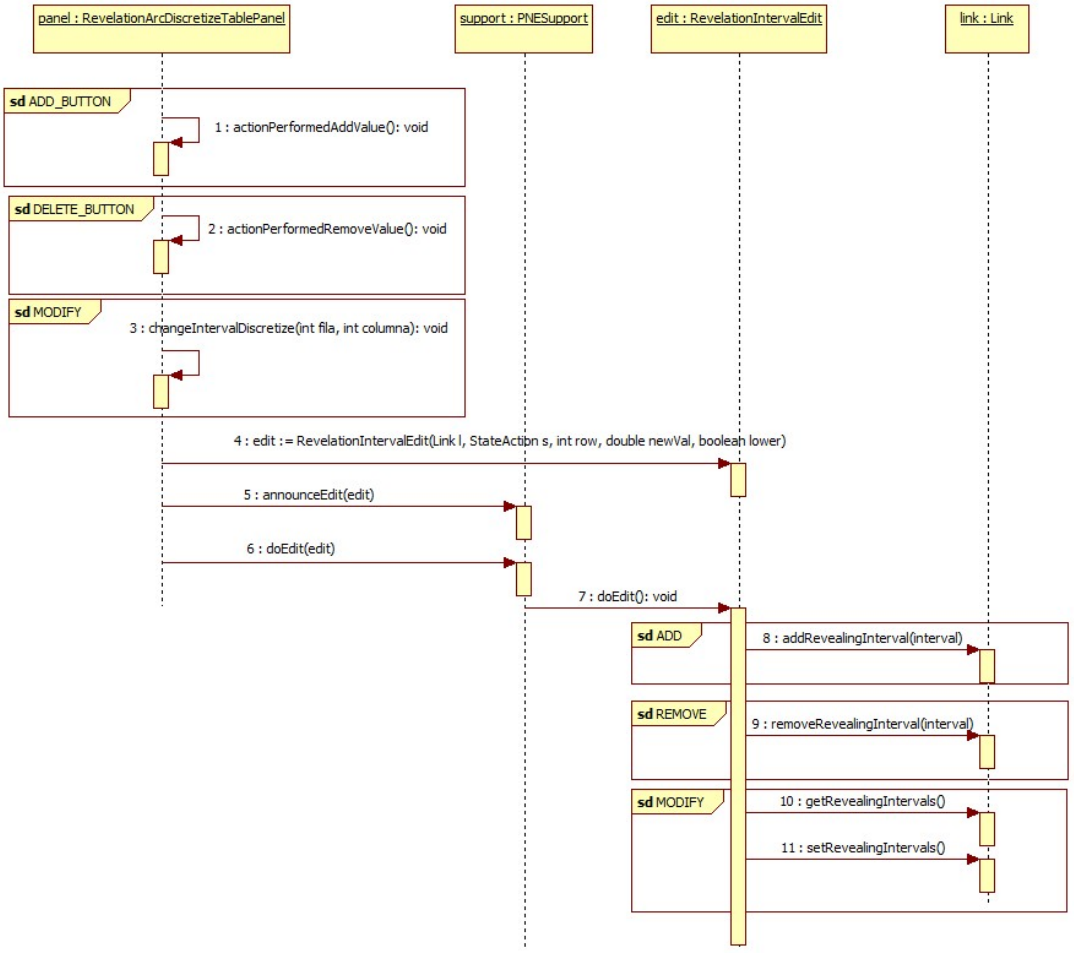


Figure 4.18: Diagram for the edition of revelation intervals

Paint revelation arc Revelation arcs are represented visually by drawing the link with a special color. The class VisualLink (package org.OpenMarkov.core.gui.graphic) needs to adapt its paint method to take into account these requirements when painting the link.

Modifications of ProbModelXML

ProbModelXML encodes the revelation conditions as the element RevelationCondition at the element Link. Figure 4.19 shows a link with revelation conditions for finite state or discretized variables, where the revelation conditions are stored as states and figure 4.20 shows a link with revelation conditions for a numeric variable. The classes PGMXReader and PGMXWriter have been adapted to implement the writing of the revelation conditions in this format.

```
<Link directed="true ">
    <Variable name="Dec:Test" />
    <Variable name="Result of test" />
    <RevelationConditions >
        <State name="do test"/>
    </RevelationConditions>
</Link >
```

Figure 4.19: ProbModelXML code for finite state revealing conditions.

```
<RevelationConditions>
    <Threshold value=number belongsTo=enumSide />2..2n
</RevelationConditions>
```

Figure 4.20: ProbModelXML code associated to numeric revealing conditions.

4.4.6 Consistence of the edition of the network

During the edition of the network the user can carry out different operations on nodes and links which may have an impact on existing link restrictions and revelation conditions, and therefore may require an additional adjustment. The basic operations such as the erasing of a link or variable remove link restrictions and revelation conditions automatically as the link is deleted from the network. Operations on the nodes of a network must be handled specifically as they require an adjustment in certain situations. The following operations on nodes imply a reset of the link restrictions on both incoming and outgoing links from a node and a reset of revelation conditions only on the outgoing links

- The edition of the type of the variable (implemented at the class VariableTypeEdit) entails a change of the domain, what may make the related link restrictions and revelation conditions inconsistent.
- The edition of the states of a finite-state variable (implemented at the classes NodeStateEdit and NodeReplaceStatesEdit) entails changes on the domain of a variable, which make the related link restrictions and revelation conditions invalid.

An example is the case where the state space of a variable A is modified by erasing a state of the variable or changing the type of the variable. In this case link restrictions and revelation conditions on the outgoing links of the node become inconsistent as they may register that a now non-existent state has a conditioning influence or a revealing influence on another variable. Therefore it is necessary to reset the link restrictions and revelation conditions on all outgoing links of the variable. Similarly the link restrictions of the incoming links of a node may reference a now non-existent state, therefore it is necessary to reset the link restrictions on the incoming

links. As revelation arcs do not establish an relation between the revealing state and the exact value which is revealed, it is not necessary to reset automatically the revelation conditions on the incoming links, because these revelation conditions do not reference the modified variable.

4.5 Codification

In this section we detail the codification of the newly implemented network constraints and the validations of the preconditions of the specific features of DANs.

Network constraints

Network constraints are implemented as realizations of the class PNConstraint, where the control of the constraint is implemented by the methods checkProbNet(ProbNet net) and checkEvent(Edit edit) (see Figure 4.3). The underlying algorithms of each constraint according to the specification in Section 4.4.1 can be found in appendix A.3. We explain the representation of the codification of the constraints by means of the example of Algorithms 1 and 2. These algorithms show the codification of the MaxNumParents constraint applied to the whole probabilistic network (checkProbNet) and applied to the edition of the network (checkEvent). The constraint MaxNumParents limits the number of parents that a node may have as specified by the argument maxNumParents. The variable constraintSatisfied represents the state of accomplishment of the constraint.

Algorithm 1 checkProbNet - MaxNumParents

Require: probNet instanceOf ProbNet, maxNumParents > 0

1: $graph \Leftarrow probNet.graph()$
2: $constraintSatisfied \Leftarrow True$
3: **for all** (*node* of graph) **do**
4: $np \Leftarrow node.numParents()$
5: **if** $(np > maxNumParents)$ **then**
6: $constraintSatisfied \Leftarrow False$
7: **end if**
8: **end for**

Algorithm 2 checkEvent - MaxNumParents

Require: edit instanceOf PNEdit, probNet instanceOf ProbNet, maxNumParents > 0

1: $constraintSatisfied \Leftarrow True$
2: **if** (*edit* instanceOf LinkEdit) **then**
3: **if** $(edit.isDirected())$ **then**
4: $node2 \Leftarrow edit.node2()$
5: $np \Leftarrow node2.numParents()$
6: **if** $(np >$ maxNumParents$)$ **then**
7: $constraintSatisfied \Leftarrow False$
8: **end if**
9: **end if**
10: **end if**

Precondition validators

The specific features of DANs, link restrictions, always observed variables and revelation arcs, are conditioned on different preconditions, as described for each case respectively in Sections 4.4.2, 4.4.4 and 4.4.5. The details of the underlying algorithms, which implement the control of the preconditions can be found at appendix A.3. We explain the approach of the validations of preconditions by means of the example of Algorithm 3. This algorithm shows the details of the implementation of the validation of the preconditions of revelation arcs, where the variable isValid represents the state of accomplishment of the precondition.

Algorithm 3 precondition - Revelation arcs

Require: link instanceOf Link, probNet instanceOf ProbNet

1: $isValid \Leftarrow True$
2: **if** (**not** probNet.hasConstraint(NoRevelationArc)) **then**
3: $node1 \Leftarrow link.node1()$
4: $node2 \Leftarrow link.node2()$
5: **if** (**not** (($node1$ typeOf Chance) **or** ($node1$ typeOf Decision)) **and** ($node2$ typeOf Chance)) **then**
6: $isValid \Leftarrow False$
7: **end if**
8: **end if**

4.6 Tests

In order to assure the quality of the software unit testing was conducted over the parts of the code, which were modified or added to OpenMarkov. OpenMarkov uses the JUnit[1] library as testing framework, which allows to control systematically that the methods of a class behave as expected. Table 4.4 shows the code coverage rate of the main packages modified for the implementation of the DAN network. The code coverage rate describes the percentage of the source code which is systematically tested with a unit test and the here presented results were measured with the software EclEmma[2], a free code coverage tool for Eclipse[3], the software development environment used for OpenMarkov.

As described with detail in appendix A.4, the implementation of link restriction and revelation arcs at the class Link is thoroughly tested; this class reaches a code coverage of 88%. Further the implementation of the new network constraints is exhaustively tested having the package org.OpenMarkov.core.model.network.constraint an overall code coverage of 69%. The new implemented network constraints go beyond this measure reaching a nearly complete code coverage, as described in appendix A.4. Also the implementation of the ProbModelXML is tested systematically having the package org.OpenMarkov.io.probmodel a coverage of 72%.

[1] http://www.junit.org/
[2] http://www.eclemma.org
[3] http://www.eclipse.org/

Modifications	package	Coverage
Link restriction/ revelation arcs	org.OpenMarkov.core.model.graph	68%
Network constraints	org.OpenMarkov.core.model.network.constraint	69%
ProbModelXML	org.OpenMarkov.io.probmodel	72%

Table 4.4: Code coverage source code

4.7 Results of implementation

This section presents the results of the modifications of OpenMarkov, in particular the new GUI elements. The screen shots are taken from the representation of a modified version of the diabetes diagnosis problem. Figure 4.21 shows the graphical model of the diabetes problem. This diagram shows the graphical representation of link restrictions, always observed variables and revelation arcs.

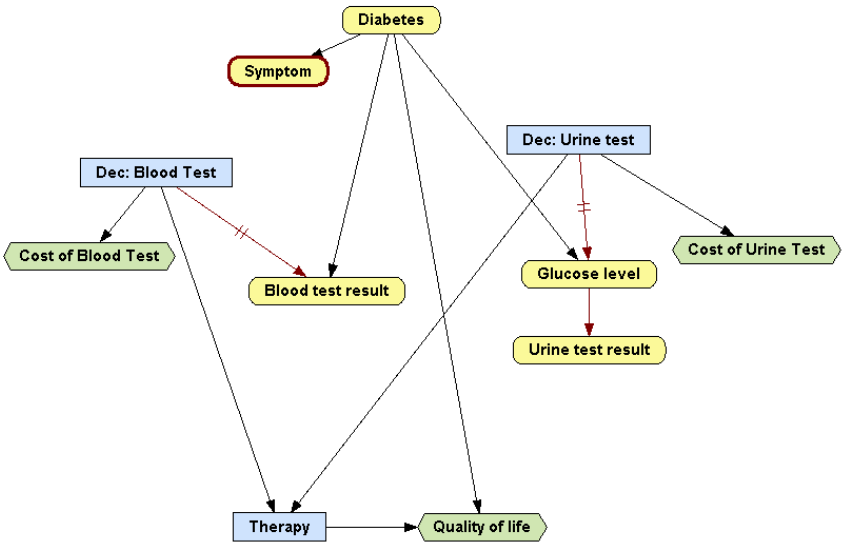


Figure 4.21: A DAN model for the diabetes problem.

Figure 4.22 shows the dialog of the edition of the link restriction. This figure shows the compatibility value table (CVT) where incompatible states are highlighted with red color and compatible states with green color.

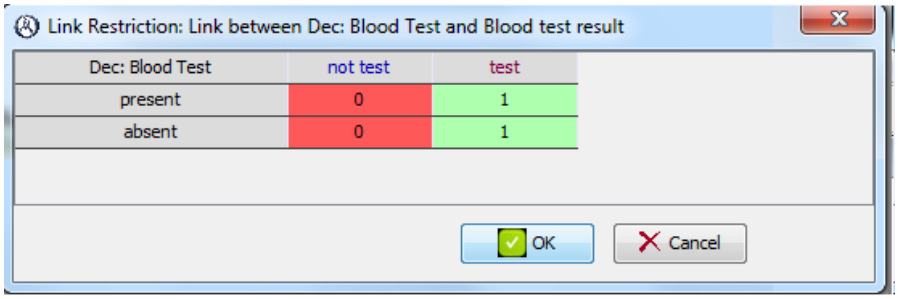


Figure 4.22: GUI for the edition of compatible states.

Figure 4.23 shows the CPT table of the *Blood test result*, where impossible states appear with red color. Figure 4.24 shows the node property pane, where the user can configure the always-observed property by means of a checkbox. Figure 4.25 shows the GUI for the configuration of the revealing states of the node *Bood test*. Figure 4.26 shows the GUI for the edition of the revealing intervals of the node *Glucose level*.

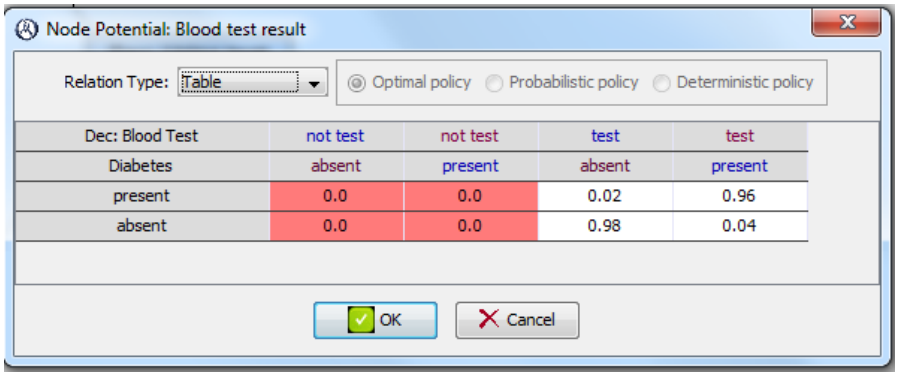


Figure 4.23: GUI edition for the edition of the conditioned probabilities.

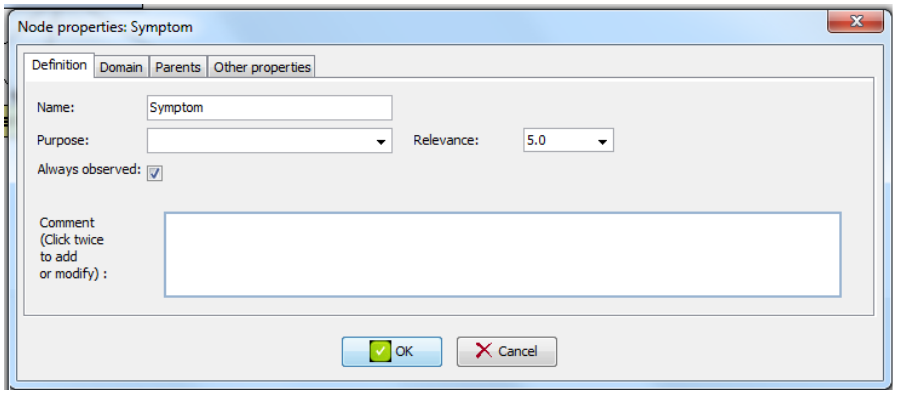


Figure 4.24: GUI for the edition of the always observed property.

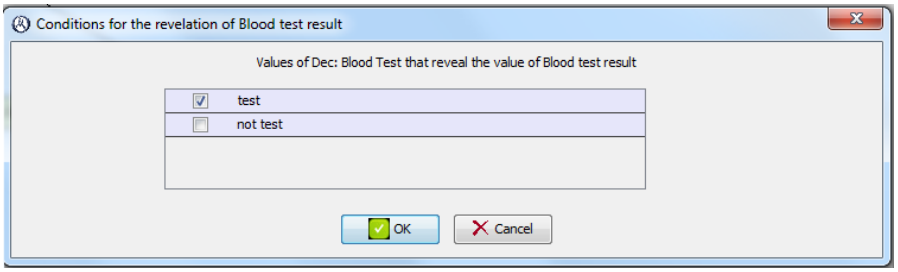


Figure 4.25: GUI for the edition for revealing states.

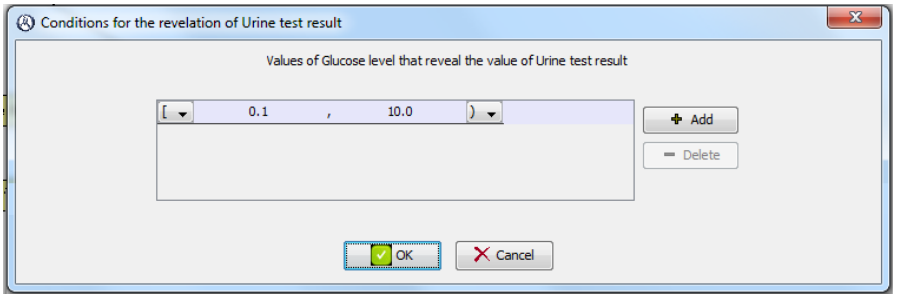


Figure 4.26: GUI for the edition of revealing intervals.

5 Conclusion

We end this dissertation by summarizing the main contributions and proposing some lines for future research.

5.1 Main contributions

First of all we have reviewed the state of the art about probabilistic graphical models (PGMs) for decision analysis in general, and the different decision analysis formalisms specific for the representation of asymmetric decision problems. We have focused in particular on the different solutions each formalism provides for the representation of order and structural asymmetry, either at the theoretical level and in practice by illustrating the solution of each formalism for three typical asymmetric decision problems often used in the literature. To our knowledge, no previous work has compared those formalisms with such a degree of detail: in general the proponents of a new formalism only discuss its applicability to one or two problems for which their method outperforms some of the methods proposed previously. Secondly we have presented decision analysis networks (DANs), which are a new graphical probabilistic model proposed by Díez & Luque (2010) intended to represent asymmetry more naturally than the previous formalisms, which were not used in practice for any real-world problem.

Another contribution is the detailed comparison of DANs with the previous formalisms, which provides a description of the strengths and weaknesses of the different formalisms. This comparison was further key to evaluate the capabilities of DANs and to assure that DANs address all types of asymmetry. For this purpose we built a DAN for each of the three sample problems used for illustration and compared the representation to the respective solutions of the alternative formalisms. During the comparison we have detected loose ends of the DAN formalism, which in close collaboration with Profs. Diez and Luque, has led to a redefinition of some of its features. In particular some aspects of the main features of DANs, restrictions and revelation arcs, were initially ambiguous or not satisfactory; they have been improved as follows:

- Restrictions have been defined in more detail, distinguishing partial and total restrictions.
- The scope of restrictions has been extended to include utility nodes, as it is useful to specify restrictions on utilities.
- The conditions under which restrictions apply have been redefined to take into account that a variable can impose a restriction only in the scenarios in which it exists.
- The conditions under which revelation arcs apply have been redefined to take into account that a chance variable can reveal other variables only when it is known, either because it was always observed or because it has been revealed in turn by other variables.

These modifications of the DAN formalism were applied in the representation of the sample problems. In consequence the revised DAN formalism now compares equally or even favorably in all important aspects with the alternative formalisms. As we have discussed in Section 3.3, all those alternative formalisms have difficulties to represent at least one of the problems introduced in chapter 2: only sequential influence diagrams (SIDs) are able to represent both order and

structural asymmetry, although asymmetric influence diagrams (AIDs) and sequential valuation networks (SVNs) resulted useful for the representation of structural asymmetry. As a result of the comparison, we have confirmed that DANs are a suitable decision analysis tool, first because DANs provide a natural representation of both order and structural asymmetry and second because DANs represent problems with local descriptions, which are independent from the complexity of the problem, what makes DANs suitable for the representation of many problems that cannot be represented efficiently with almost all of the alternative formalisms.

Finally another important contributions of this work was the implementation of DANs in OpenMarkov, an open-source software tool for the edition and evaluation of PGMs with the objective that DANs can be used in practice for decision analysis. The details of the software implementation are explained in chapter 4.

5.2 Future work

There are some open lines for future research, one regarding the development of the formalism and the second regarding the practical application.

With respect to the development and implementation of DANs, the only evaluation method proposed so far is the evaluation of the equivalent decision tree, which is not implemented yet. Díez & Luque (2010) mention that it would be desirable to implement a more efficient evaluation method for DANs. For example one line of research would be to adapt the algorithm by Demirer & Shenoy (2006), which decomposes a sequential valuation network (SVN) into a set of completely ordered sub-problems. Another possibility would be to adapt the algorithm for unconstrained influence diagrams (UIDs), which is based on the construction of S-DAGs (Ahlmann-Ohlsen et al., 2009; Luque et al., 2010).

With respect to decision analysis it would be interesting to build a DAN for several real-world problems in order to evaluate its suitability. One possibility would be to build DANs for some medical problems that have been analyzed previously with influence diagrams, such as IctNeo (Bielza et al., 1999) for neonated jaundice, Mediastinet (Luque 2009) for the mediastinal staging of non-small cell lung cancer, and ArthroNet (León 2011) for total knee arthroplasty. The comparison between these IDs and the corresponding DAN would bring new light on the advantages and shortcomings of each formalism. Another line for future work would be to extend this comparison also to sequential influence diagrams (SIDs), which are the most important competitors of DANs at the present. Both formalisms are able to represent structural and order asymmetry efficiently, although their approaches to represent asymmetry are different. It would be interesting to investigate which kind of problems are better representable with DANs or SIDs. In this work we have described at the theoretical level briefly some characteristics of decision problem we consider responsible for making a problem better representable with one formalism than another. It would be interesting to check this theory at the practice analyzing different real-world problems to see if they are better representable with DANs or SIDs.

A Appendices

A.1 Probability computations for the equivalent decision tree

This section describes the computations of the conditioned probabilities for the upper branch of the equivalent decision tree for the diabetes problem shown in chapter 3.3. The conditional probabilities are calculated from the joint probabilities of the scenarios using marginalization.

1. Resolve scenarios ($(+s, +bt, +b, +ut, +u, +d)$ and $(+s, +bt, +b, +ut, +u, \neg d)$:

   (a) Determine joint probabilities for the scenarios:
   $P(+s, bt, +bt, +b, +ut, +u, +tr, +d) = P(+s| + d) \cdot P(+b| + d, +bt) \cdot P(+u| + d, +ut) \cdot P(+d)$
   $P(+s, bt, +bt, +b, +ut, +u, +tr, \neg d) = P(+s|\neg d) \cdot P(+b|\neg d, +bt) \cdot P(+u|\neg d, +ut) \cdot P(\neg d)$
   $P(+s, bt, +bt, +b, +ut, +u, +tr) = 0,0554 + 1,86 \cdot 10^{-7}$
   The probability distributions $P(s|d), P(b|bt, d), P(u|ut, d)$ and $P(d)$ are available from the DAN and used to calculate the probabilities of the final scenarios.

   (b) Determine the conditioned probability of the branches:

   $P(+d| + s, bt, +bt, +b, +ut, +u, +tr) = \frac{P(+s,bt,+bt,+b,+ut,+u,+tr,+d)}{P(+s,bt,+bt,+b,+ut,+u,+tr)} = 0,99$
   $P(\neg d| + s, bt, +bt, +b, +ut, +u, +tr) = \frac{P(+s,bt+bt,+b,+ut,+u,+tr\neg d)}{P(+s,bt,+bt,+b,+ut,+u,+tr)} = \frac{1,86 \cdot 10^{-7}}{0,0554} = 3,357 \cdot 10^{-6}$

2. Resolve scenarios $(+s, bt, +bt, +b, +u)$ and $(+s, bt, +bt, +b, \neg u)$:

   (a) Determine joint probabilities for the scenarios:

   $P(+s, bt + bt, +b, +ut, +u) = P(+s) \cdot P(+b| + bt) \cdot P(+u| + ut) = 0,0004$
   $P(+s, bt, +bt, +b, +ut, \neg u) = P(+s) \cdot P(+b| + bt) \cdot P(\neg u| + ut) = 0,00474$
   The probability distributions $P(s), P(b|bt)$ and $P(u|ut)$ used to calculate the probabilities of the scenarios are not directly available from the DAN. It is necessary to obtain these new probability distributions by marginalizing the now irrelevant variable d:
   $P(s) = \sum_d P(s|d) \cdot P(d)$
   $P(b|bt) = \sum_d P(b|bt, d) \cdot P(d)$
   $P(u|ut) = \sum_d P(u|ut, d) \cdot P(d)$

   (b) Determine the conditioned probability of the branches:
   $P(+u| + s, bt, +bt, +b, +ut) = \frac{P(+s,bt,+bt,+b,+ut,+u)}{P(+s,bt,+bt,+b,+ut)} = \frac{0,0004}{0,00514} = 0,077$
   $P(\neg u| + s, bt, +bt, +b, +ut) = \frac{P(+s,bt,+bt,+b,+ut,\neg u)}{P(+s,bt,+bt,+b,+ut)} = 0,922$

3. Resolve scenarios $(+s, bt, +bt, +b))$ and $(+s, bt, +bt, \neg b))$

(a) Determine joint probabilities for the scenarios:

$P(+s, bt, +bt, +b) = P(s) \cdot P(+b| + bt) = 0,00514$

$P(+s, bt, +bt, \neg b) = P(s) \cdot P(\neg b| + bt) = 0,05523$

(b) Determine the conditioned probability of the branches:

$P(+b| + s, bt, +bt) = \frac{P(+s,+bt,+b)}{P(+s,+bt)} = 0,085$

$P(\neg b| + s, bt, +bt) = \frac{P(+s,+bt,\neg b)}{P(+s,+bt)} = 0,915$

4. Resolve scenarios $(+s)$ and $(\neg s)$
The probability distribution of S is $P(+s)$ =0,064 and $P(\neg s)$ =0,94.

A.2 Use case description

This appendix contains the detailed description of the use cases shown in Table 4.1.

Use Case 1	**Create Decision Analysis Network**
Primary Actor:	End-user
Description:	Creates a new decision analysis network.
Preconditions:	
Postconditions:	A new instance of a DAN exists.
Related Use Cases:	

Main Success Scenario:

1. The user selects the option 'New network' from the menu.
2. The system opens a dialog showing the available network types.
3. The user selects the network type 'Decision analysis Network' and confirms clicking the 'Accept' button.
4. The system hides the dialog, creates a new instance of a decision analysis network and activates at the GUI of Openmarkov the correspondent options for the edition of a decision analysis network.

Extensions:

Step 3: The user selects the button 'Cancel' and the system hides the dialog without creating a new network.

Use Case 2	**Store DAN**
Primary Actor:	End-user
Description:	Stores the DAN in a pgmx file.
Preconditions:	The current network is a DAN.
Postconditions:	The configuration of the DAN is stored in a pgmx file.
Related Use Cases:	

Main Success Scenario:

1. The user selects the option 'Save as' from the menu.
2. The system opens a file chooser dialog, where the user can introduce the file name and location.
3. After intoducing the file name and the location, the user selects the option 'Save'.
4. The system hides the file chooser dialog and stores the configuration of the DAN in a file in format ProbModelXML and having the extension 'pgmx'.

Extensions:

Step 3: The user selects the 'Cancel' button and the system hides the dialog without storing the file.

Use Case 3	**Read DAN**
Primary Actor:	End-user
Description:	Reads a DAN from a pgmx file and represents it graphically.
Preconditions:	
Postconditions:	The GUI of Openmarkov shows the graphical model of the DAN
Related Use Cases:	Use case 4: Represent graphical model

Main Success Scenario:

1. The user selects the option 'Open' from the menu.
2. The system opens a file chooser dialog where the user can select the pgmx file of the DAN.
3. After selecting a pgmx file, the user confirms clicking the button 'Open' of the dialog.
4. The system hides the file chooser dialog and reads the configuration of the DAN from the pgmx file.
5. The system shows the graphical model of the DAN at the GUI (Use case 4 'Represent graphical model).

Extensions:

Step 3: The user clicks the button 'Cancel' and the system hides the file chooser dialog whithout reading a file.

Step 5: The configuration of the DAN of the pgmx is incorrect and the system shows an error message.

Use Case 4	**Represent graphical model**
Primary Actor:	GUI
Description:	Represent the graphical model of a DAN including special features such as link restrictions, revelation arcs and always observed variables.
Preconditions:	The network is a DAN.
Postconditions:	The GUI of Openmarkov shows the graphical model of the DAN.
Related Use Cases:	◦ Use case 5: Paint link restriction. ◦ Use case 6: Paint revelation arc. ◦ Use case 7: Paint always observed variable.

Main Success Scenario:

1. The GUI paints the graphical model of a DAN.
2. For each link having a link restriction it starts the Use case 5: Paint link restriction.
3. For each link having revelation conditions it starts the Use case 6: Paint revelation arc.
4. For each variable, which is always observed it starts the Use case 7: Paint always observed variable.

Extensions:

Use Case 5	**Paint link restriction**
Primary Actor:	End-user,GUI
Description:	Represents graphically a link restriction.
Preconditions:	The network is a DAN and a link restriction exists.
Postconditions:	The GUI of Openmarkov shows the link restriction explicitly.
Related Use Cases:	Use case 4: Represent graphical model

Main Success Scenario:

1. The GUI paints the graphical model of a DAN.
2. The GUI paints all links with a link restriction with a double stripe crossing the link if the link restriction is total and with a single stripe if the link restriction is partial (See Figure A.1).
3. If the user selects the link, the link and the double stripe are highlighted painting broader lines.

Extensions:

Figure A.1: Graphical representation of a link restriction.

Use Case 6	**Paint revelation arc**
Primary Actor:	End-user,GUI
Description:	Represents graphically a revelation arc.
Preconditions:	The network is a DAN and a revelation arc exists.
Postconditions:	The GUI of Openmarkov shows revelation arcs explicitly.
Related Use Cases:	Use case 4: Represent graphical model

Main Success Scenario:

1. The GUI paints the graphical model of a DAN.
2. The GUI paints all links with revelation conditions with a special color.
3. If the user selects a link with a revelation condition, the link is highlighted painting it with a broader line.

Extensions:

Use Case 7	**Paint always observed variables**
Primary Actor:	End-user,GUI
Description:	Represents graphically an always observed variable.
Preconditions:	The network is a DAN and an always observed variables exists.
Postconditions:	The GUI of Openmarkov shows always observed variables explicitly.
Related Use Cases:	Use case 4: Represent graphical model

Main Success Scenario:

1. The GUI paints the graphical model of a DAN.
2. The GUI paints with a special color the border of all variable, which are always observed.
3. If the user selects an always observed node, its boarder is painted wider.

Extensions:

Use Case 8	**Edition link restrictions**
Primary Actor:	End-user
Description:	Assign compatibility values to the states of the variables of a link.
Preconditions:	◦ The network is a DAN and a link is selected. ◦ The first node of the link is a decision or chance variable.
Postconditions:	The system stores the compatibility values set by the user as link restrictions related to the link.
Related Use Cases:	

Main Success Scenario:

1. The user selects the option 'Edit link restrictions' or 'Add link restrictions' from the contextual menu of a link.
2. The system shows a dialog for the edition of the compatibility values of the link restriction, which is similar to the scheme shown at Figure A.2 for a directed link between A and B. This dialog lets the user add or remove restrictions between the values of variables A and B. By default the restrictions do not exist and the table contains the value 1 for every pair of values (a,b). The user can pick a cell of the table and by a double mouse click change its value. The admitted values are 1 and 0, which represent compatibility and incompatibility, respectively. Compatible combinations are depicted with a clear green color and incompatible values are depicted with a dark red color.
3. After adding or removing the restrictions, the user confirms the edition by clicking the 'Accept' button .
4. The system stores the compatibility values of the states according to the changes made by the user in a link restriction, which is related to the link.

Extensions:

Step 3: The user clicks the 'Cancel' button. The system hides the dialog without storing the edited values.

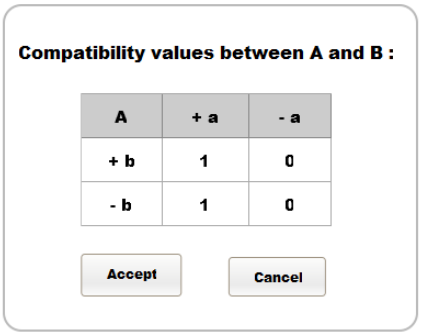


Figure A.2: Scheme of the GUI for the edition of link restrictions.

Use Case 9	**Remove link restriction**
Primary Actor:	End-user
Description:	Removes the link restriction from the selected link.
Preconditions:	◦ The network is a DAN. ◦ A link with a link restriction is selected.
Postconditions:	The link has no link restriction.
Related Use Cases:	
Main Success Scenario: 1. The user selects the option 'Remove link restriction' from the contextual menu of the link. 2. The system removes the link restriction from the link.	
Extensions:	

Use Case 10	**Represent incompatible states at CTP**
Primary Actor:	End-user
Description:	Depict visually incompatible states at the CPT.
Preconditions:	◦ The network is a DAN and a link is selected. ◦ A chance node has a link restriction on an incoming link.
Postconditions:	
Related Use Cases:	
Main Success Scenario: 1. The user opens the dialog for the edition of the CPT of the chance node. 2. The system shows the conditioned probability table and the states which are impossible because of a link restriction are highlighted with a special color. These cells are not-editable and have probability zero.	
Extensions:	

Use Case 11	**Configure always observed property**
Primary Actor:	End-user
Description:	Configuration of the always observed property.
Preconditions:	◦ The network is a DAN. ◦ A chance node is selected.
Postconditions:	The system stores the always observed property according to the configuration of the user
Related Use Cases:	

Main Success Scenario:

1. The user opens the node property dialog of the selected node.
2. The system shows a checkbox for the always observed property at the 'Definition' tab.
3. After setting or unsetting this property, the user confirms the edition clicking the 'Accept' button.
4. The system closes the node properties dialog and stores the property for the node.

Extensions:

Step 3: The user clicks the 'Cancel' button and the system hides the dialog without storing the property.

Use Case 12	**Define revealing conditions**
Primary Actor:	End-user
Description:	Defines the set of revealing conditions for a link.
Preconditions:	◦ The network is a DAN. ◦ A link is selected and the first node is chance or decision and the second node is chance.
Postconditions:	The system stores the revelation conditions of the link.
Related Use Cases:	◦ Use case 13: Define revealing states ◦ Use case 14: Define revealing conditions

Main Success Scenario:

1. The end-user selects the option 'Edit revelation conditions' from the contextual menu of the link.
2. Depending on the domain of the first node, the edition of the revealing values is accomplished in two different ways:

   Assignment of revealing values for finite state or discretized variables (Use case 13 'Define revealing states').

   Assignment of revealing values of numeric variables (Use case 14 'Define revealing intervals').

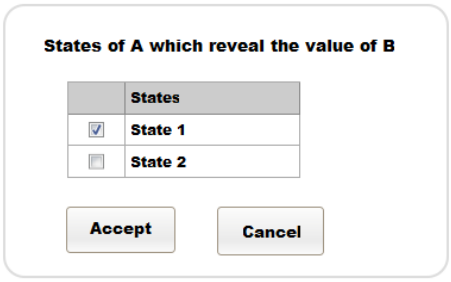


Figure A.3: Edition of revealing states.

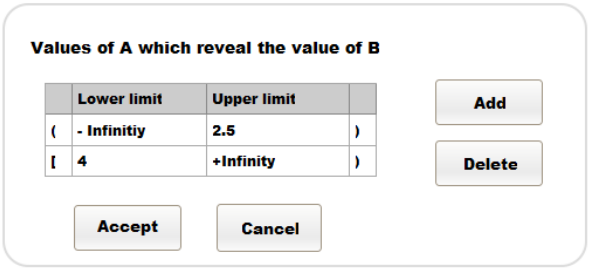


Figure A.4: Edition of revealing intervals.

Use Case 13	**Define revealing states**
Primary Actor:	End-user
Description:	Defines the states which reveal the value of a node.
Preconditions:	◦ The network is a DAN. ◦ A link is selected and the first node of the link is a finite state or discretized variable.
Postconditions:	The system stores the revelation states of the link.
Related Use Cases:	Use case 12: Define revealing conditions

Main Success Scenario:

1. The system shows a dialog for the configuration of the revealing states, which is similar to the scheme shown in Figure A.3. This window lets the user select from a list the states of the variable A which reveal the value of B. Beside each state appears a checkbox for adding that state to the list of revealing states or removing it from the list.
2. After assigning the revealing conditions of the variable A, the user confirms the configuration by clicking the 'Accept' button.
3. The revealing conditions dialog disappears and the revealing states are stored at the system.

Extensions:

Step 2: The uer clicks the 'Cancel' button and the dialog disappears without saving the current configuration of the revelation states.

Use Case 14	**Define revealing intervals**
Primary Actor:	End-user
Description:	Defines the intervals which reveal the value of a node.
Preconditions:	◦ The network is a DAN ◦ A link is selected and the first node is a variable with numeric domain
Postconditions:	The system stores the revelation states of the link.
Related Use Cases:	Use case 12: Define revealing conditions

Main Success Scenario:

1. The system shows the graphical window for the configuration of the revealing intervals, which is similar to the scheme shown in Figure A.4, where the user can specify the intervals of A, which reveal the values of B. The user can add and delete intervals by using the 'Add' and 'Delete' button or modify the existing intervals clicking at the cell of the interval and editing the values of the interval.
2. After assigning the revealing intervals of the variable , the user confirms the configuration by clicking the 'Accept' button.
3. The revealing conditions dialog disappears and the revealing intervals are stored at the system.

Extensions:

Step 2: The uer clicks the 'Cancel' button and the dialog disappears without saving the current configuration of the revelation intervals.

A.3 Detailed description of the algorithms

In the following we detail the underlying algorithms of the network constraints according to their specification at section4.4.1. The constraints NoRevelationArc and NoRestriction are not shown as they have no algorithmic content:

Algorithm 4 checkProbNet - NoClosedPath

Require: probNet instanceOf ProbNet
1: $graph \Leftarrow probNet.graph()$
2: $constraintSatisfied \Leftarrow True$
3: $noLoops \Leftarrow NoLoopsConstraint.checkProbnet(probNet)$
4: $noCycles \Leftarrow NoCyclesConstraint.checkProbnet(probNet)$
5: **if not** ($noLoops$ **and** $noCycles$) **then**
6: $constraintSatisfied \Leftarrow False$
7: **end if**

Algorithm 5 checkEvent - NoClosedPath

Require: edit instanceOf PNEdit, probNet instanceOf ProbNet
1: $constraintSatisfied \Leftarrow True$
2: $noLoops \Leftarrow NoLoopsConstraint.checkEdit(probNet, edit)$
3: $noCycles \Leftarrow NoCyclesConstraint.checkEdit(probNet, edit)$
4: **if not** ($noLoops$ **and** $noCycles$) **then**
5: $constraintSatisfied \Leftarrow False$
6: **end if**

Algorithm 6 checkProbNet - NoMixedParents

Require: probNet instanceOf ProbNet
1: $graph \Leftarrow probNet.graph()$
2: $constraintSatisfied \Leftarrow True$
3: **for all** ($node$ of graph) **do**
4: **if** ($node$ typeOf Utility) **then**
5: $utilityParent \Leftarrow False$
6: $chanceDecisionParent \Leftarrow False$
7: **for all** (parentNode of node.parents()) **do**
8: **if** ($parent$ typeOf Utility) **then**
9: $utilityParent \Leftarrow True$
10: **end if**
11: **if** (($parent$ typeOf Chance) **or** ($parent$ typeOf Decision)) **then**
12: $chanceDecisionParent \Leftarrow True$
13: **end if**
14: **if** ($utilityParent$ **and** $chanceDecisionParent$) **then**
15: $constraintSatisfied \Leftarrow False$
16: **end if**
17: **end for**
18: **end if**
19: **end for**

The codification of the NoMixedParents constraint at the routine checkEvent() is based on the code of line 4-18 of algorithm 6.

Algorithm 7 checkProbNet - NoMultipleLinks

Require: probNet instanceOf ProbNet

```
1:  graph ⇐ probNet.graph()
2:  constraintSatisfied ⇐ True
3:  for all (node of graph) do
4:    for all (link of node.links()) do
5:      node1 ⇐ link.node1()
6:      node2 ⇐ link.node2()
7:      directed ⇐ link.isDirected()
8:      if directed then
9:        if (graph.existsLink(node1, node2, undirected)) then
10:         constraintSatisfied ⇐ False
11:       end if
12:     else
13:       if (graph.existsLink(node1, node2, directed)) or
          (graph.existsLink(node2, node1, directed)) then
14:         constraintSatisfied ⇐ False
15:       end if
16:     end if
17:   end for
18: end for
```

The codification of the NoMultipleLinks constraint at the routine checkEvent() is based on the code of line 5-16 of algorithm 7.

Algorithm 8 checkProbNet - NoUtilityParent

Require: probNet instanceOf ProbNet

```
1:  graph ⇐ probNet.graph()
2:  constraintSatisfied ⇐ True
3:  for all (node of graph) do
4:    if (node typeOf Utility) then
5:      for all (childNode of node.children()) do
6:        if (childNode not typeOf Utility) then
7:          constraintSatisfied ⇐ False
8:        end if
9:      end for
10:   end if
11: end for
```

The codification of the NoUtilityParent constraint at the routine checkEvent() is based on the code of line 4-10 of algorithm 8.

Algorithm 9 checkProbNet - DistinctLinks

Require: probNet instanceOf ProbNet
1: $graph \Leftarrow probNet.graph()$
2: $constraintSatisfied \Leftarrow True$
3: **for all** (node of graph) **do**
4: $nl \Leftarrow node.numLinks()$
5: $np \Leftarrow node.numParents()$
6: $nc \Leftarrow node.numChildren()$
7: $ns \Leftarrow node.numSiblings()$
8: **if** $nl > (nc + np + ns)$ **then**
9: $constraintSatisfied \Leftarrow False$
10: **end if**
11: **end for**

Algorithm 10 checkEvent - DistinctLinks

Require: edit instanceOf PNEdit, probNet instanceOf ProbNet
1: $graph \Leftarrow probNet.graph()$
2: $constraintSatisfied \Leftarrow True$
3: **if** (*edit* instanceOf LinkEdit) **then**
4: $node1 \Leftarrow edit.node1()$
5: $node2 \Leftarrow edit.node2()$
6: $link \Leftarrow edit.link()$
7: $directed \Leftarrow link.isDirected()$
8: **if** *directed* **then**
9: **if** $(graph.existsLink(node1, node2, directed))$ **then**
10: $constraintSatisfied \Leftarrow False$
11: **end if**
12: **else**
13: **if** $(graph.existsLink(node1, node2, undirected))$ **or** $(graph.existsLink(node2, node1, undirected))$ **then**
14: $constraintSatisfied \Leftarrow False$
15: **end if**
16: **end if**
17: **end if**

Algorithm 11 checkProbNet - OnlyAtemporalVariables

Require: probNet instanceOf ProbNet
1: $constraintSatisfied \Leftarrow True$
2: **for all** (*variable* of probNet) **do**
3: **if** (*variable* typeOf Temporal) **then**
4: $constraintSatisfied \Leftarrow False$
5: **end if**
6: **end for**

Algorithm 12 checkEvent - OnlyAtemporalVariables

Require: edit instanceOf PNEdit
  $constraintSatisfied \Leftarrow True$
  if (*edit* instanceOf ProbNodeEdit) **then**
    $variable \Leftarrow edit.variable()$
    if (*variable* typeOf Temporal) **then**
      $constraintSatisfied \Leftarrow False$
    end if
  end if

Algorithm 13 checkProbNet - OnlyFiniteStateVariables

Require: probNet instanceOf ProbNet
1: $constraintSatisfied \Leftarrow True$
2: **for all** (*variable* of probNet) **do**
3: $node \Leftarrow variable.getNode()$
4: **if** (*node* typeOf Chance)**or** (*node* typeOf Decision) **then**
5: **if** ((*variable* **not** typeOf FiniteState) **or**
  (*variable* **not** typeOf Discretized)) **then**
6: $constraintSatisfied \Leftarrow False$
7: **end if**
8: **end if**
9: **end for**

The codification of the OnlyFiniteStateVariable constraint at the routine checkEvent() is based on the code of line 3-8 of algorithm 13.

Algorithm 14 checkProbNet - OnlyNumericVariables

Require: probNet instanceOf ProbNet
1: $constraintSatisfied \Leftarrow True$
2: **for all** (*variable* of probNet) **do**
3: **if** (*variable* **not** typeOf Numeric) **then**
4: $constraintSatisfied \Leftarrow False$
5: **end if**
6: **end for**

The codification of the OnlyNumericVariable constraint at the routine checkEvent() is based on the code of line 3-5 of algorithm 14.

Algorithm 15 checkProbNet - OnlyTemporalVariables

Require: probNet instanceOf ProbNet
1: $constraintSatisfied \Leftarrow True$
2: **for all** (*variable* of probNet) **do**
3: **if** (*variable* **not** typeOf Temporal) **then**
4: $constraintSatisfied \Leftarrow False$
5: **end if**
6: **end for**

The codification of the OnlyTemporalVariable constraint at the routine checkEvent() is based on the code of line 3-5 of algorithm 15.

Validators

In the following we detail the algorithms of the control of the preconditions of link restrictions and always observed variables according to their specification at section 4.4.2 and 4.4.4 respectively:

Algorithm 16 precondition - Link restrictions

Require: link instanceOf Link, probNet instanceOf ProbNet

```
1:  isValid ⇐ True
2:  if (not probNet.hasConstraint(NoRestriction)) then
3:     node1 ⇐ link.node1()
4:     node2 ⇐ link.node2()
5:     if ( (node1 typeOf Chance) or (node1 typeOf Decision)) then
6:        var1 ⇐ node1.variable()
7:        var2 ⇐ node2.variable()
8:        if ( (var1 not typeOf FiniteStates) or
          (((node2 not typeOf Utility) and (var2 not typeOf FiniteState)) then
9:           isValid ⇐ False
10:       end if
11:    else
12:       isValid ⇐ False
13:    end if
14: else
15:    isValid ⇐ False
16: end if
```

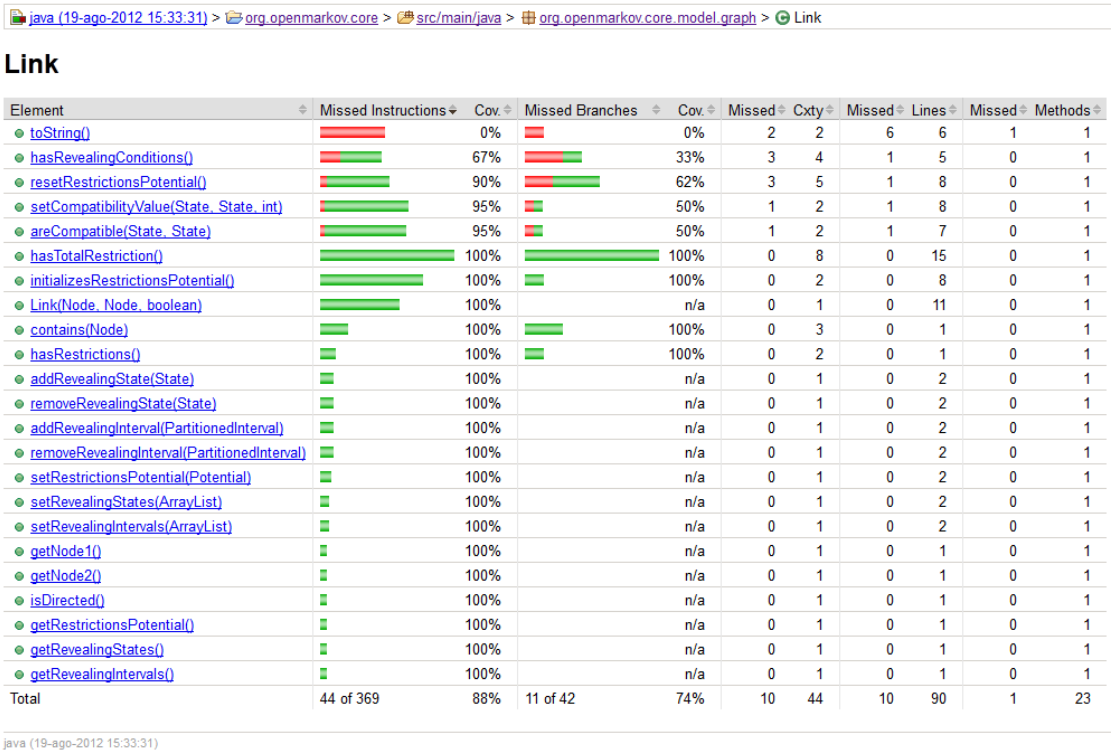
Algorithm 17 precondition - Always observed variable

Require: node instanceOf Node, probNet instanceOf ProbNet

```
1:  isValid ⇐ True
2:  if (not probNet.hasConstraint(NoRevelationArc)) then
3:     if (not ((node typeOf Chance)) then
4:        isValid ⇐ False
5:     end if
6:  end if
```

A.4 Detailed description of the code coverage

This appendix shows the code coverage obtained from unit testing related to the implementations of the new network constraints, the modifications of the data model and ProbModelXML. These results have been measured with the software EclEmma[1], a free code coverage tool for Eclipse[2], the software development environment used for OpenMarkov. Figure A.5 shows that the class Link, where link restrictions and revelation arcs are implemented, has a code coverage of 88,1%. Figure A.6 shows the results of unit testing of the network constraints. Although the overall code coverage of the package is only 69%, the new implemented network constraints have a nearly complete code coverage. Figure A.7 shows the details of unit testing for the implementation of ProbModelXML. This package has a code coverage of 72% by testing the reading and writing in format ProbModelXML systematically.


java (19-ago-2012 15:33:31) > org.openmarkov.core > src/main/java > org.openmarkov.core.model.graph > Link

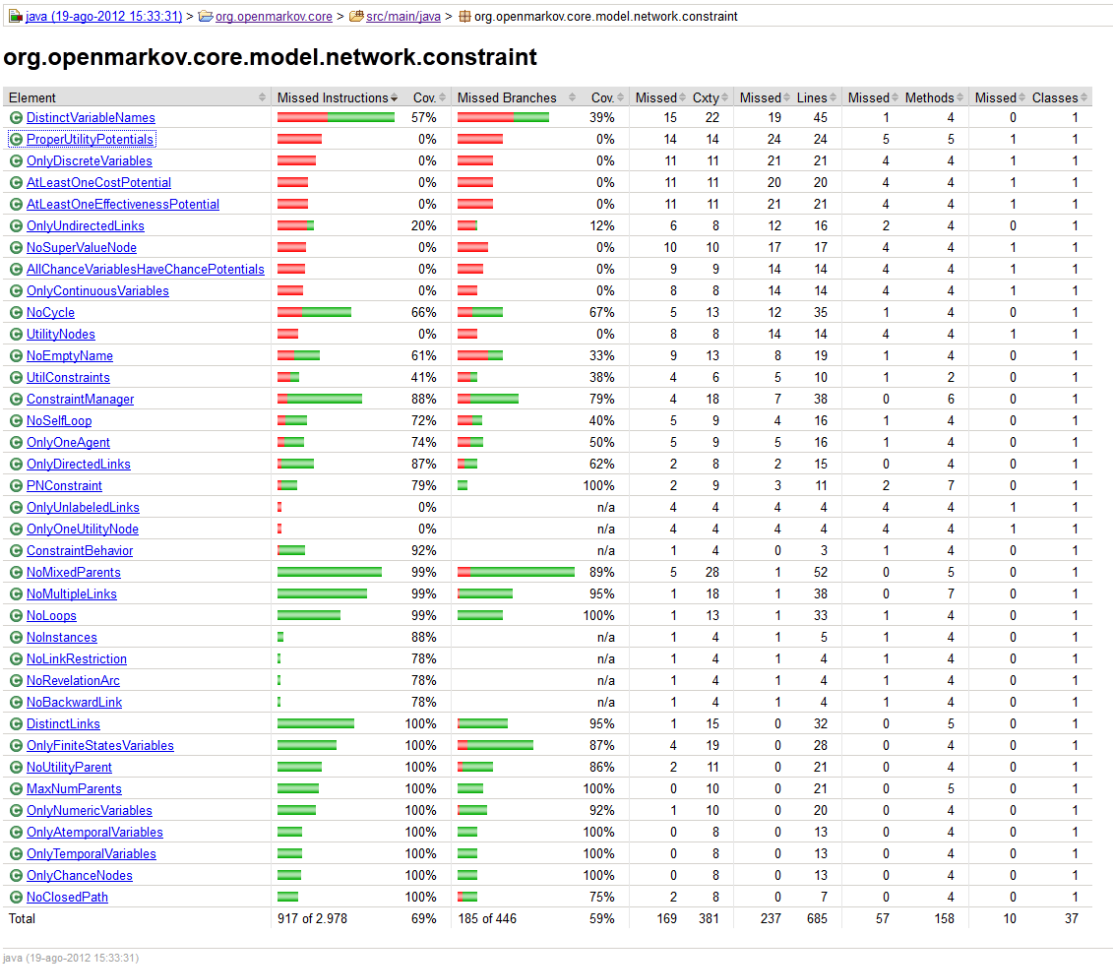
Link

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
toString()		0%		0%	2	2	6	6	1	1
hasRevealingConditions()		67%		33%	3	4	1	5	0	1
resetRestrictionsPotential()		90%		62%	3	5	1	8	0	1
setCompatibilityValue(State, State, int)		95%		50%	1	2	1	8	0	1
areCompatible(State, State)		95%		50%	1	2	1	7	0	1
hasTotalRestriction()		100%		100%	0	8	0	15	0	1
initializesRestrictionsPotential()		100%		100%	0	2	0	8	0	1
Link(Node, Node, boolean)		100%		n/a	0	1	0	11	0	1
contains(Node)		100%		100%	0	3	0	1	0	1
hasRestrictions()		100%		100%	0	2	0	1	0	1
addRevealingState(State)		100%		n/a	0	1	0	2	0	1
removeRevealingState(State)		100%		n/a	0	1	0	2	0	1
addRevealingInterval(PartitionedInterval)		100%		n/a	0	1	0	2	0	1
removeRevealingInterval(PartitionedInterval)		100%		n/a	0	1	0	2	0	1
setRestrictionsPotential(Potential)		100%		n/a	0	1	0	2	0	1
setRevealingStates(ArrayList)		100%		n/a	0	1	0	2	0	1
setRevealingIntervals(ArrayList)		100%		n/a	0	1	0	2	0	1
getNode1()		100%		n/a	0	1	0	1	0	1
getNode2()		100%		n/a	0	1	0	1	0	1
isDirected()		100%		n/a	0	1	0	1	0	1
getRestrictionsPotential()		100%		n/a	0	1	0	1	0	1
getRevealingStates()		100%		n/a	0	1	0	1	0	1
getRevealingIntervals()		100%		n/a	0	1	0	1	0	1
Total	44 of 369	88%	11 of 42	74%	10	44	10	90	1	23

java (19-ago-2012 15:33:31)

Figure A.5: Code coverage rates of the class Link.

[1] http://www.eclemma.org
[2] http://www.eclipse.org/



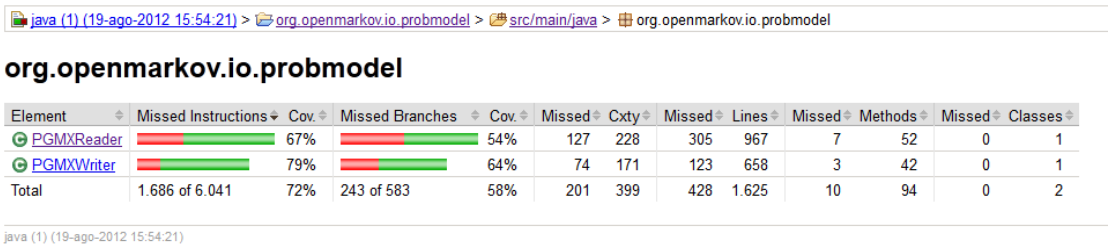
java (19-ago-2012 15:33:31) > org.openmarkov.core > src/main/java > org.openmarkov.core.model.network.constraint

org.openmarkov.core.model.network.constraint

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
DistinctVariableNames		57%		39%	15	22	19	45	1	4	0	1
ProperUtilityPotentials		0%		0%	14	14	24	24	5	5	1	1
OnlyDiscreteVariables		0%		0%	11	11	21	21	4	4	1	1
AtLeastOneCostPotential		0%		0%	11	11	20	20	4	4	1	1
AtLeastOneEffectivenessPotential		0%		0%	11	11	21	21	4	4	1	1
OnlyUndirectedLinks		20%		12%	6	8	12	16	2	4	0	1
NoSuperValueNode		0%		0%	10	10	17	17	4	4	1	1
AllChanceVariablesHaveChancePotentials		0%		0%	9	9	14	14	4	4	1	1
OnlyContinuousVariables		0%		0%	8	8	14	14	4	4	1	1
NoCycle		66%		67%	5	13	12	35	1	4	0	1
UtilityNodes		0%		0%	8	8	14	14	4	4	1	1
NoEmptyName		61%		33%	9	13	8	19	1	4	0	1
UtilConstraints		41%		38%	4	6	5	10	1	2	0	1
ConstraintManager		88%		79%	4	18	7	38	0	6	0	1
NoSelfLoop		72%		40%	5	9	4	16	1	4	0	1
OnlyOneAgent		74%		50%	5	9	5	16	1	4	0	1
OnlyDirectedLinks		87%		62%	2	8	2	15	0	4	0	1
PNConstraint		79%		100%	2	9	3	11	2	7	0	1
OnlyUnlabeledLinks		0%		n/a	4	4	4	4	4	4	1	1
OnlyOneUtilityNode		0%		n/a	4	4	4	4	4	4	1	1
ConstraintBehavior		92%		n/a	1	4	0	3	1	4	0	1
NoMixedParents		99%		89%	5	28	1	52	0	5	0	1
NoMultipleLinks		99%		95%	1	18	1	38	0	7	0	1
NoLoops		99%		100%	1	13	1	33	1	4	0	1
NoInstances		88%		n/a	1	4	1	5	1	4	0	1
NoLinkRestriction		78%		n/a	1	4	1	4	1	4	0	1
NoRevelationArc		78%		n/a	1	4	1	4	1	4	0	1
NoBackwardLink		78%		n/a	1	4	1	4	1	4	0	1
DistinctLinks		100%		95%	1	15	0	32	0	5	0	1
OnlyFiniteStatesVariables		100%		87%	4	19	0	28	0	4	0	1
NoUtilityParent		100%		86%	2	11	0	21	0	4	0	1
MaxNumParents		100%		100%	0	10	0	21	0	5	0	1
OnlyNumericVariables		100%		92%	1	10	0	20	0	4	0	1
OnlyAtemporalVariables		100%		100%	0	8	0	13	0	4	0	1
OnlyTemporalVariables		100%		100%	0	8	0	13	0	4	0	1
OnlyChanceNodes		100%		100%	0	8	0	13	0	4	0	1
NoClosedPath		100%		75%	2	8	0	7	0	4	0	1
Total	917 of 2.978	69%	185 of 446	59%	169	381	237	685	57	158	10	37

java (19-ago-2012 15:33:31)

Figure A.6: Code coverage rates of the network constraints.



java (1) (19-ago-2012 15:54:21) > org.openmarkov.io.probmodel > src/main/java > org.openmarkov.io.probmodel

org.openmarkov.io.probmodel

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
PGMXReader		67%		54%	127	228	305	967	7	52	0	1
PGMXWriter		79%		64%	74	171	123	658	3	42	0	1
Total	1.686 of 6.041	72%	243 of 583	58%	201	399	428	1.625	10	94	0	2

java (1) (19-ago-2012 15:54:21)

Figure A.7: Code coverage rate of the package ProbModelXML.

A.5 Resumen en español (Summary in spanish)

Motivación

La construcción de sistemas inteligentes para la toma de decisiones con incertidumbre es uno de los objetivos principales de la Inteligencia Artificial (IA), que ha sido abordada por diferentes paradigmas. Desde los inicios de la IA hasta los años 1990 gran expectativas han sido puestos en sistemas experto basados en reglas, pero debido a su especificidad para un dominio y la ineficiencia de cálculos distribuidos durante la inferencia, estos sistemas son ahora menos relevante. En cambio, modelos probabilistas gráficos (MPGs), en particular redes bayesianas y diagramas de influencia, están ganando cada vez más importancia desde sus inicios a principios de los años 1980. MPGs tratan la incertidumbre aplicando la teoría de probabilidad bayesiana, lo que les permite realizar inferencia de una forma eficiente y resolver problemas con una complejidad que no han podido ser abordados con otras técnicas conocidas hasta este momento. Además MPGs tienen un modelo gráfico que codifica las relaciones de dependencia directamente en un grafo creando así una descripción cualitativa y fácil de entender del problema de decisión.

Aunque los diagramas de influencia proporcionan una solución eficiente para la representación del conocimiento y razonamiento, estos modelos tienen dificultades para representar problemas de decisiones asimétricas, es decir, situaciones donde el valor de una variable o las opciones de decisión son restringidos por observaciones o decisiones previas (asimetría estructural) o el orden de las decisiones no está definido (asimetría de orden). Dado que los problemas reales suelen ser muchas veces asimétricos, existe una necesidad de encontrar formalismos de análisis de decisiones que son capaces de representar asimetría. Varios formalismos específicos han sido propuestos en las últimas décadas, pero ninguno de ellos ha sido utilizado para construir una aplicación de un problema real, lo que puede significar que no son lo suficientemente simple para facilitar la construcción del modelo o la comunicación con el experto. Este último aspecto es muy importante en campos como la medicina, donde el experto necesita entender el razonamiento del sistema para aceptar el sistema experto. Por estas razones Díez & Luque (2010) han propuesto un formalismo nuevo, los redes de análisis de decisiones (RADs), que representan asimetría más natural. Hasta esta tesis de máster, los RADs han sido presentados como un nuevo formalismo con una definición teórica del modelo y un método de evaluación, ambos ilustrados mediante un ejemplo simple y comparado brevemente con las soluciones alternativas propuestas hasta el momento.

La motivación principal de este trabajo ha sido ampliar la investigación de Díez & Luque (2010) sobre RADs. Primero era necesario realizar una comparación más detallada de RADs con los formalismos alternativos para asegurar que los RADs abordan todos los aspectos de la representación de asimetría. Esta tesis se centra en analizar las capacidades de modelación de los RADs mediante la representación de tres problemas de decisión asimétricos que aparecen en la literatura y que contienen todos los tipos de asimetría. La representación de los RADs es comparada con las soluciones de los alternativos formalismos y esta comparación debería hacer visible los puntos fuertes o débiles de los RADs y permitir mejorar el formalismo si fuera necesario. En segundo lugar, era necesario implementar los RADs en una herramienta de ayuda a la toma de decisiones para que el nuevo formalismo puede ser utilizado en la práctica. Hemos implementado los RADs en OpenMarkov, una herramienta de software libre para la edición y evaluación de modelos gráficos probabilistas desarrollado por el CISIAD [3] en la Universidad Nacional de Educación a Distancia. Esta herramienta implementa ya diferentes redes probabilistas para el análisis de decisiones y es disponible para cualquier persona, dado que es un proyecto libre y de código abierto que se distribuye gratuitamente.

[3] CISIAD significa Centro de Investigación sobre Sistemas Inteligentes de Ayuda a la Decisión

Objetivos

Debido a las necesidades descritas en la sección previa, los objetivos de esta investigación pueden resumirse en:

1. Analizar el formalismo de los RADs para asegurar que desde el punto de vista de sintaxis y semántica los RADs resuelven todos los aspectos relevantes de la representación de asimetría correctamente:

   (a) Comparar la capacidad de los RADs en referencia a la representación de todos los tipos de asimetría con la de otros formalismos de análisis de decisiones.

   (b) Revisar el uso de las propiedades principales de los RADs, como el orden temporal parcial, restricciones y arcos de revelación en la representación de problemas de decisión para refinar su especificación. Esta tarea incluye la revisión del significado de las restricciones, el alcance y condiciones bajo las que restricciones y arcos de revelación son efectivos y el análisis del uso de arcos de revelación para describir precedencia de información.

2. Implementar RADs en OpenMarkov para que sea posible usarlos en la práctica para representar problemas de decisiones.

Metodología

El trabajo presentado en esta tesis es de carácter analítico. El objetivo es asegurar que el formalismo de los RADs represente asimetría de una forma completa y correcta. La metodología seguida para alcanzar estos objetivos se basa en un análisis de las capacidades de los RADs para la representación de aspectos asimétricos y una comparación critica de la solución de los RADs con otros formalismos. Esta metodología es un proceso iterativo, que involucra cuatro fases tal como está descrita en la Figura A.8:

1. Definición de las características del formalismo.

2. Implementación de los RADs en OpenMarkov.

3. Representación de problemas de decisión asimétricos con RADs.

4. Comparación de la representación de problemas asimétricos con RADs respecto a la de otros formalismos.

A partir de la definición inicial de los RADs de Díez & Luque (2010), la primera fase es la definición de las características del formalismo de los RADs. La segunda fase es la implementación de esta definición en OpenMarkov, lo que posibilita la representación de diferentes problemas asimétricos con RADs en la tercera fase. En esta fase utilizamos diferentes problemas de decisión asimétricos que aparece en la literatura. La representación con RADs de cada problema de decisión es comparada en la siguiente fase con la correspondiente solución de cada uno de los otros formalismos. Esta fase incluye eventualmente la revisión de soluciones ya publicadas en la literatura. Como esta comparación puede revelar puntos fuertes y débiles de los RADs, el proceso es iniciado otra vez para mejorar el formalismo de los RADs.

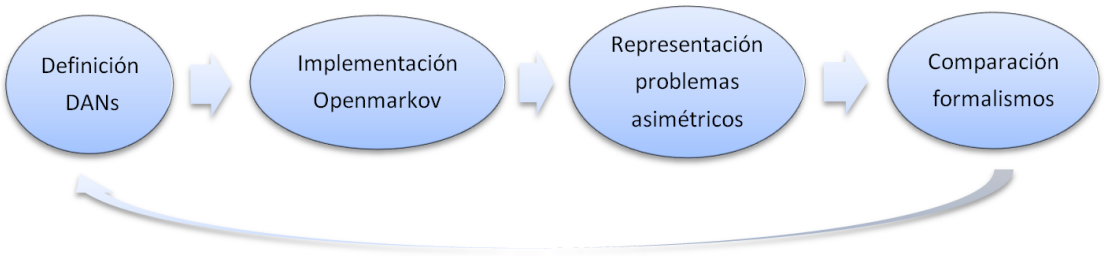


Figura A.8: Fases en el desarrollo del análisis.

Organización de la tesis

Esta tesis del máster está estructurada en cinco capítulos. El capitulo 1 presenta la motivación, los objetivos y la metodología seguida para este trabajo de investigación. El capitulo 2 revisa el estado de conocimiento de los formalismos de análisis de decisiones en general y en especifico para la representación de problemas de decisiones asimétricos. Esta parte presenta una introducción a los formalismos de análisis de decisiones, una descripción de los fundamentos de MGPs y una discusión sobre el estado de varios formalismos alternativos conocidos hasta el momento para la representación de problemas de decisión asimétricos. Los alternativos formalismos analizados son árboles de decisión, diagramas de influencia, diagramas de influencia extendidos, redes de valuación secuenciales, diagramas de influencia asimétricos, diagramas de influencia sin restricciones y diagramas de influencia secuenciales. El capitulo 3 presenta los RADs y explica sus propiedades para la representación de asimetría. La representación de los problemas de decisión asimétricos es utilizada para una comparación de las ventajas e limitaciones de los RADs respecto a los formalismos alternativos. El capitulo 4 explica la implementación de los RADs en OpenMarkov describiendo las nuevas propiedades del sistema informático. El Capitulo 5 presenta la conclusión y algunas líneas de investigación para el futuro.

Principales contribuciones

A principio de este trabajo hemos revisado el estado de arte de los modelos probabilistas gráficos (MPGs) para el análisis de decisiones en general, y los diferentes formalismos específicos para la representación de problemas de decisión asimétricos. Nos hemos centrado en particular en los diferentes soluciones que cada formalismo proporciona para la representación de la asimetría de orden y estructural, tanto a nivel teórico y en práctica ilustrando las correspondientes soluciones de cada formalismo con la representación de tres problemas asimétricos que aparecen en la literatura. Según que sepamos, ningún trabajo anterior ha comparado estos formalismos con tanto detalle: en general los proponentes de un nuevo formalismo solamente discutan su aplicabilidad para uno o dos problemas para las cuales su método resulta superior a los métodos propuestos anteriormente. En segundo lugar hemos presentado los redes de análisis de decisiones (RADs), que son un nuevo modelo probabilista gráfico propuesto por Díez & Luque (2010) con la intención de representar asimetría con más naturalidad que los formalismos previos, que además no han sido utilizados en la practica para ningún problema real.

Otra contribución es una comparación detallada de los RADs con los formalismos previos, que proporcionan una descripción de las ventajas e limitaciones de los diferentes formalismos. Esta comparación ha sido además clave para evaluar la capacidad de los RADs y asegurar que los RADs abordan la representación de todos los tipos de asimetría. Para este fin hemos construido

una RAD para cada una de las tres ejemplos ilustrativos y hemos comparado esta representación con las respectivas soluciones de los formalismos alternativos. Durante esta comparación hemos detectado algunos cabos sueltos del formalismo de los RADs, lo que ha llevado, en colaboración con los profesores Diez y Luque, a una redefinición de algunas de las propiedades de los RADs. En concreto algunos aspectos de las propiedades principales de los RADs, restricciones y arcos de revelación, han sido inicialmente ambiguos o no satisfactorios; estas propiedades han sido mejoradas de la siguiente forma:

- Las restricciones han sido definidos en más detalle, distinguiendo restricciones parciales y totales.
- El alcance de las restricciones ha sido ampliado para incluir nodos de utilidad, porque es conveniente especificar restricciones sobre utilidades.
- Las condiciones bajo las cuales restricciones son efectivas han sido redefinidas para considerar que una variable puede imponer restricciones solamente si existe en un escenario.
- Las condiciones bajo las cuales arcos de revelación son efectivas han sido redefinidas para considerar que una variable aleatoria puede revelar otras variables solamente cuando es conocida, bien porque es siempre observada o porque ha sido revelada a su vez por otras variables.

Estas modificaciones de los RADs ha sido aplicada en la representación de los problemas de ejemplo. En consecuencia el formalismo de los RADs revisado es ahora equiparable o incluso mejor que los otros formalismos en todos los aspectos importantes. Tal como hemos comentado en la Sección 3.3, todos estos formalismos alternativos tienen dificultades para representar como mínimo uno de los problemas introducidos en el capitulo 2: sólo diagramas de influencia secuenciales son capaces de representar a la vez asimetría de orden y estructura, aunque diagramas de influencia asimétricas y redes de valoración secuenciales han resultado útiles para la representación de asimetría estructural. Como resultado de la comparación podemos confirmar que RADs como herramienta de análisis de decisiones son útiles, primero porque RADs proporcionan una representación natural de la asimetría de orden y estructural y en segundo lugar porque RADs representan problemas con una descripción local, que es independiente de la complejidad del problema, lo que hace RADs apropiadas para la representación de muchos problemas que no pueden ser representados eficientemente con la gran mayoría de los formalismos alternativos.

Finalmente otra contribución importante de este trabajo ha sido la implementación de RADs en OpenMarkov, un programa libre para la edición y evaluación de MPGs, con el objetivo que RADs puedan ser utilizados en práctica para el análisis de decisiones. Los detalles de la implementación están explicados en el capitulo 4.

Futuro trabajo

Existen diferentes líneas para futuro trabajo - una relacionada con el desarrollo del formalismo y una segunda acerca de la aplicación practica de los RADs. Con respecto al desarrollo y la implementación de los RADs, el único método de evaluación propuesto hasta el momento es la evaluación del árbol de decisión equivalente, lo que no esta implementado todavía. Díez & Luque (2010) mencionan que seria interesante implementar un método de evaluación más eficiente para los RADs. Por ejemplo una linea de trabajo seria la adaptación del algoritmo de Demirer & Shenoy (2006), que descompone una red de valuación secuencial en un conjunto completamente ordenado de subproblemas. Otra posibilidad podría ser la adaptación del algoritmo

para diagramas de influencia sin restricciones, que esta basado en la construcción de S-DAGs (Ahlmann-Ohlsen et al., 2009; Luque et al., 2010).

En referencia al análisis de decisiones seria interesante construir una RAD para varios problemas reales con el fin de evaluar su aptitud. Una posibilidad seria construir RADs para problemas médicos que han sido analizados anteriormente con diagramas de influencia, como por ejemplo IctNeo (Bielza et al., 1999) para la icteria del recién nacido, Mediastinet (Luque 2009) para la estadificación mediastinal de cáncer de pulmón, y ArthroNet (León 2011) para la artroplastia total de rodilla. La comparación entre estos diagramas de influencia y la correspondiente RAD aportaría nuevo conocimiento sobre las ventajas y limitaciones de cada formalismo. Otra linea de trabajo futuro seria la extensión de esta comparación también a diagramas de influencia secuenciales, que son los competidores más importantes de los RADs en la actualidad. Ambos formalismos pueden representar tanto asimetría de orden y estructural eficientemente, aunque los métodos de representación de asimetría son diferentes. Seria interesante investigar que tipos de problemas pueden ser representados mejor con RADs o con SIDs. En este trabajo hemos descrito a un nivel teórico brevemente algunas características de problemas de decisión que creemos que hacen que sean mejor representados por un formalismo que otro. Seria interesante contrastar esta teoría en la practica analizando diferentes problemas reales para averiguar si son mejor representables con RADs o con SIDs.

Bibliography

Ahlmann-Ohlsen, K. S., Jensen, F. V., Nielsen, T. D., Pedersen, O., & Vomlelová, M. (2009). A comparison of two approaches for solving unconstrained influence diagrams. *International Journal of Approximate Reasoning*, 50, 153–173.

Arias, M. (2009). *Carmen: Una Herramienta de Software Libre para Modelos Gráficos Probabilistas*. PhD thesis, UNED, Madrid. In Spanish.

Arias, M. & Díez, F. J. (2008). Carmen: An open source project for probabilistic graphical models. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM'08)* (pp. 25–32). Hirtshals, Denmark.

Arias, M., Díez, F. J., & Palacios, M. P. (2010). *OpenMarkovXML. A format for encoding probabilistic graphical models*. Technical Report CISIAD-10-04, UNED, Madrid, Spain.

Arias, M., Díez, F. J., & Palacios, M. P. (2011). *ProbModelXML. A format for encoding probabilistic graphical models*. Technical Report CISIAD-11-02, UNED, Madrid, Spain.

Bayes, T. & R.Price (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, 53, 370–418.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk. *Econometrica*, 22, 23–36.

Bielza, C., Gómez, M., & Shenoy, P. P. (2011). A review of representation issues and modelling challenges with influence diagrams. *Omega*, 39, 227–241.

Bielza, C., Ríos, S., & Gómez, M. (1999). Influence diagrams for neonatal jaundice management. In *AIMDM'99: Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making* (pp. 138–142). London, UK: Springer-Verlag.

Bielza, C. & Shenoy, P. P. (1999). A comparison of graphical techniques for asymmetric decision problems. *Management Science*, 45, 1552–1569.

Call, H. & Miller, W. (1990). A comparison of approaches and implementations for automating decision analysis. *Reliability Engineering and System Safety*, 30, 115–162.

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In *Proceedings of the Proceedings of the Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-88)* (pp. 55–63). New York, NY.

Cooper, G. F. & Herskovits, E. (1991). A Bayesian method for constructing Bayesian belief networks from databases. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence (UAI'91)* (pp. 86–94). Los Angeles, CA: Morgan Kaufmann, San Mateo, CA.

Covaliu, Z. & Oliver, R. M. (1995). Representation and solution of decision problems using sequential decision diagrams. *Management Science*, 41, 1860–1881.

Demirer, R. & Shenoy, P. P. (2006). Sequential valuation networks for asymmetric decision problems. *European Journal of Operational Research*, 169, 286–309.

Díez, F. J. & Luque, M. (2010). *Representing decision problems with Decision Analysis Networks.* Technical Report CISIAD-10-01, UNED, Madrid, Spain.

Díez, F. J., Palacios, M. A., & Arias, M. (2011). MDPs in medicine: Opportunities and challenges. In *Decision Making in Partially Observable, Uncertain Worlds: Exploring Insights from Multiple Communities (IJCAI Workshop)* Barcelona, Spain.

Díez, F. J. & van Gerven, M. A. J. (2011). Dynamic LIMIDs. In L. E. Sucar, J. Hoey, & E. Morales (Eds.), *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions* (pp. 164–189). Hershey, PA: IGI Global.

Fung, R. M. & Shachter, R. D. (1990). *Contingent influence diagrams.* Technical Report, Engineering-Economic Systems Department, Stanford University, Stanford, CA.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes.* Cambridge, MA: The MIT Press.

Howard, R. A. (1966). Decision analysis: Applied decision theory. In *Proceedings of the Fourth International Conference on Operational Research* (pp. 55–71). New York: Wiley-Interscience.

Howard, R. A. (1988). Decision analysis: Practice and promise. *Management Science*, 34, 679–695.

Howard, R. A. & Matheson, J. E. (1984). Influence diagrams. In R. A. Howard & J. E. Matheson (Eds.), *Readings on the Principles and Applications of Decision Analysis* (pp. 719–762). Menlo Park, CA: Strategic Decisions Group.

Jensen, F. V., Nielsen, T. D., & Shenoy, P. (2006). Sequential influence diagrams: A unified asymmetry framework. *International Journal of Approximate Reasoning*, 42, 101–118.

Jensen, F. V., Olesen, K. G., & Andersen, S. K. (1990). An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20, 637–660.

Jensen, F. V. & Vomlelová, M. (2002). Unconstrained influence diagrams. In *Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI'02)* (pp. 234–241). San Francisco, CA: Morgan Kaufmann.

Kim, J. H. & Pearl, J. (1983). A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI–83)* (pp. 190–193). Karlsruhe, Germany.

Kjærulff, U. & Madsen, A. L. (2010). *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis.* New York: Springer.

Kuhn, H. W. (1953). *Extensive games and the problem of information.* Princeton, NJ: Princeton University Press.

Lacave, C., Luque, M., & Díez, F. J. (2007). Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37, 952–965.

Laplace, P. (1812). *Theorie analytique des probabilites.* Courcier.

Lauritzen, S. L. & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50, 157–224.

León, D. (2011). A probabilistic graphical model for total knee arthroplasty. Master's thesis, Dept. Artificial Intelligence, UNED, Madrid, Spain.

Luque, M. (2009). *Probabilistic Graphical Models for Decision Making in Medicine*. PhD thesis, UNED, Madrid.

Luque, M. & Díez, F. J. (2007). Representación de problemas de decisión con asimetrías de orden. In *Actas del II Simposio de Inteligencia Computacional* Zaragoza, Spain.

Luque, M. & Díez, F. J. (2010). Variable elimination for influence diagrams with super-value nodes. *International Journal of Approximate Reasoning*, 51, 615 – 631.

Luque, M., Díez, F. J., & Disdier, C. (2009). A decision support system for the mediastinal staging of non-small cell lung cancer. In *31st Annual Meeting of the Society for Medical Decision Making* Los Angeles, CA.

Luque, M., Nielsen, T. D., & Jensen, F. V. (2010). *An anytime algorithm for evaluating unconstrained influence diagrams*. Technical Report CISIAD-10-05, Dept. Artificial Intelligence, UNED, Madrid, Spain.

Nielsen, T. D. (2001). *Graphical Models for Partially Sequential Decision Problems*. PhD thesis, Dept. Computer Science, Aalborg University.

Nielsen, T. D. & Jensen, F. (2000). Representing and solving asymmetric Bayesian decision problems. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)* (pp. 416–425). San Francisco, CA: Morgan Kaufmann.

Nielsen, T. D. & Jensen, F. V. (1999a). *Representing and solving asymmetric Bayesian decision problems*. Technical report, Dept. Computer Science, Aalborg University.

Nielsen, T. D. & Jensen, F. V. (1999b). Welldefined decision scenarios. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)* (pp. 502–511). San Francisco, CA: Morgan Kaufmann.

Olmsted, S. M. (1983). *On Representing and Solving Decision Problems*. PhD thesis, Dept. Engineering-Economic Systems, Stanford University, CA.

Pauker, S. & Wong, J. (2005). The influence of influence diagrams in medicine. *Decision Analysis*, 2, 238–244.

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29, 241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Pearl, J. (1993). Belief networks revisited. *Artificial Intelligence*, 59, 49–56.

Pearl, J., Geiger, D., & Verma, T. (1989). Conditional independence and its representations. *Kybernetika*, 25, 33–44.

Qi, R., Zhang, N. L., & Poole, D. (1994). Solving asymmetric decision problems with influence diagrams. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)* (pp. 491–497). San Francisco, CA: Morgan Kaufmann.

Raiffa, H. (1968). *Decision Analysis. Introductory Lectures on Choices under Uncertainty.* Reading, MA: Addison-Wesley.

Raiffa, H. & Schlaifer, R. (1961). *Applied Statistical Decision Theory.* Cambridge: MIT Press.

Savage, L. J. (1954). *The Foundations of Statistics.* New York: Wiley.

Shachter, R. & Peot, M. (1992). Decision making using probabilistic inference methods. In *Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence (UAI-92)* (pp. 276–28). San Mateo, CA: Morgan Kaufmann.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, 34, 871–882.

Shachter, R. D. (1988). Probabilistic inference and influence diagrams. *Operations Research*, 36, 589–605.

Shachter, R. D. (1998). Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)* (pp. 480–487). San Francisco, CA: Morgan Kaufmann.

Shenoy, P. P. (1992). Valuation based systems for Bayesian decision analysis. *Operations Research*, 40, 463–484.

Shenoy, P. P. (1994). A comparison of graphical techniques for decision analysis. *European Journal of Operational Research*, 78, 1–21.

Shenoy, P. P. (1996). Representing and solving asymmetric decision problems using valuation networks. *Lecture Notes in Statistics*, 112, 99–108.

Shenoy, P. P. (1998). Game trees for decision analysis. *Theory and Decision*, 44, 149–171.

Shenoy, P. P. (2000). Valuation network representation and solution of asymmetric decision problems. *European Journal of Operational Research*, 121, 579–608.

Smith, J. E., Holtzman, S., & Matheson, J. E. (1993). Structuring conditional relationships in influence diagrams. *Operations Research*, 41, 280–297.

Spiegelhalter, D. J. & Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20, 579–605.

Tatman, J. A. & Shachter, R. D. (1990). Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20, 365–379.

von Neumann, J. & Morgenstern, O. (1944). *Theory of Games and Economic Behavior.* Princeton, NJ: Princeton University Press.

Wald, A. (1939). Contributions to the theory of statistical estimation and testing hypotheses. *Ann. Math. Statistics*, 10, 299–326.

Wald, A. (1950). *Statistical decision theory.* New York: McGraw-Hill.

Wright, S. (1921). Correlation and causation. *Journal of Agricultural Research*, 20, 557–585.

Zhang, N. L., Qi, R., & Poole, D. (1994). A computational theory of decision networks. *International Journal of Approximate Reasoning*, 11, 83–158.