UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

BERNARDO MARTINS DA LUZ

Bayesian BDI Agents and Approaches to Desire Selection

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Rosa Vicari
Advisor

Porto Alegre, August 2013

CIP – CATALOGING-IN-PUBLICATION

Luz, Bernardo Martins da

Bayesian BDI Agents and
Approaches to Desire Selection / Bernardo Martins da Luz. – Porto Alegre: PPGC da UFRGS, 2013.

118 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Rosa Vicari.

1. Selection. 2. Desires. 3. Nondeterministic. 4. Desire selection. 5. Uncertainty. 6. Bayes. 7. Bayesian Network. 8. Agent. 9. BDI agent. 10. Bias. I. Vicari, Rosa. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Pró-Reitor de Coordenação Acadêmica: Vice-Reitor Rui Vicente Oppermann
Pró-Reitora de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

EPIGRAPH

"Miracles aren't meant to be wished for, they're meant to be created by our own power."

– Norihiro Yagi, *Miria* in *Claymore*, chapter 127

DEDICATORY

This endeavor could not have been finished without support from different sources. Far too long has been the journey, and far too great – overwhelmingly so – the psychological burden that yielded this dissertation. There is no appropriate or sufficient means of expressing the nature of this experience.

This work would not have been possible were it not for the support of my parents, Clésis Vânia Jorge Martins and Álvaro Ricardo Wilke da Luz. The circumstances would simply not have allowed it otherwise.

Thanks to my cat, Bino (in memoriam), and my grandmothers, Eneida Jorge Martins (in memoriam) and Sybila Pereira Wilke (in memoriam), for supporting me prior to and throughout this work.

Thanks to my advisor, Rosa Maria Vicari, for being patient, providing support and allowing perseverance. Thanks, as well, to Felipe Rech Meneguzzi, for sparing the time to provide very appreciated input for this dissertation.

It would probably be a mistake to name friends who provided support, but did not actually provide objective input on the subject of this dissertation, as the list would likely end up regrettably incomplete, not to mention inappropriately long for this section. If you are reading this, you can tell whether or not you have had an impact. ;)

Bernardo

"People partake of your life in a multitude of ways.
Imbued with their own individual weight and influence.
Their influence is assimilated into your life's story.
Their presence also endures or not in a multitude of manners.
When it does not, a multitude of durations mark their respective leaves.
As do a multitude of circumstances and a multitude of reasons.
Their role and relevance is forever engraved into you in the context of existence itself.
All is carried along the flow of time. And the flow of life."

AGRADECIMENTOS

Esta empreitada não teria sido finalizada sem o suporte de diferentes fontes. Demasiadamente longa foi a jornada, e demasiadamente grande a carga psicológica que gerou esta dissertação. Não há maneira apropriada ou suficiente de expressar a natureza desta experiência.

Este trabalho não teria sido possível sem o suporte dos meus pais, Clésis Vânia Jorge Martins e Álvaro Ricardo Wilke da Luz. As circunstâncias simplesmente não o teriam permitido de outro modo.

Obrigado ao meu gato, Bino (in memoriam), e as minhas avós, Eneida Jorge Martins (in memoriam) e Sybila Pereira Wilke (in memoriam), por me darem suporte antes e durante este trabalho.

Obrigado a minha orientadora, Rosa Maria Vicari, por ser paciente, prover suporte e permitir perseverança. Obrigado, também, a Felipe Rech Meneguzzi, por ter contribuído a esta dissertação.

Provavelmente seria um erro citar amigos que proveram suporte, mas não diretamente contribuíram com o conteúdo desta dissertação, já que a lista provavelmente ficaria lamentavelmente incompleta, sem mencionar inapropriadamente longa para esta seção. Se você está lendo isto, você pode dizer se teve ou não impacto. ;)

Bernardo

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
BDI	Belief-Desire-Intention
DAG	Directed Acyclic Graph

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The reasoning performed in BDI agents essentially involves manipulating three data structures representing their beliefs, desires and intentions. Traditional BDI agents' beliefs do not represent uncertainty, and may be expressed as a closed set of ground literals. The constraints that indicate whether a given desire is viable and passive to be adopted as an intention in traditional BDI agents may be represented as logical expressions over beliefs.

Given that Bayesian Networks allow one to represent uncertain information probabilistically, Bayesian BDI agents employ Bayesian Networks to support uncertainty in their beliefs. In Bayesian BDI agents, beliefs represented in Bayesian Networks refer to states of event variables, holding individual dynamic probabilities that account for the uncertainty. The processes that constitute reasoning in this agent model require changes in order to accomodate this difference. Among these processes, this work is specifically concerned with desire selection.

A previous strategy for desire selection is based on applying a threshold on belief probabilities. However, such an approach precludes an agent from selecting desires conditioned on beliefs with probabilities below a certain threshold, even if those desires could be achieved if they were selected. To address this limitation, we develop three alternative approaches to desire selection under uncertainty: Probability Ranking, Biased Lottery and Multi-Desire Biased Random Selection. Probability Ranking selects a desire using a list of desires sorted in decreasing order of precondition probability. Biased Lottery selects a desire using one random numeric value and desire-associated numeric intervals proportional to the probabilities of the desires' preconditions. Multi-Desire Biased Random Selection selects multiple desires using random numeric values and considering the probabilities of their preconditions.

We present examples, including the *Watchman* agent, as well as experiments involving the latter, to show how these approaches allow an agent to sometimes select desires whose belief preconditions have very low probabilities.

Keywords: Selection, desires, nondeterministic, desire selection, uncertainty, Bayes, Bayesian Network, agent, BDI agent, bias.

Agentes BDI Bayesianos e Abordagens para Seleção de Desejos

RESUMO

O raciocínio realizado em agentes BDI envolve essencialmente manipular três estruturas de dados representando suas crenças, desejos e intenções. Crenças de agentes BDI tradicionais não representam incerteza, e podem ser expressas como um conjunto fechado de literais *ground*. As restrições que indicam se um dado desejo é viável e pode ser adotado como uma intenção em agentes BDI tradicionais podem ser representadas como expressões lógicas sobre crenças.

Dado que Redes Bayesianas permitem que representem-se informações com incerteza probabilisticamente, agentes BDI bayesianos as empregam para suportar incerteza em suas crenças. Em agentes BDI bayesianos, crenças representadas em Redes Bayesianas referem-se a estados de variáveis de eventos, possuindo probabilidades dinâmicas individuais que referem-se à incerteza. Os processos the constituem o raciocínio neste modelo de agente requerem mudanças a fim de acomodar esta diferença. Dentre estes processos, este trabalho concentra-se especificamente na seleção de desejos.

Uma estratégia prévia para seleção de desejos é baseada em aplicar um limiar a probabilidades de crenças. Entretanto, tal abordagem impede que um agente selecione desejos condicionados em crenças cujas probabilidades estejam abaixo de um certo limiar, mesmo que tais desejos pudessem ser atingidos caso fossem selecionados. Para lidar com esta limitação, desenvolvemos três abordagens alternativas para seleção de desejos sob incerteza: Ranking Probabilístico, Loteria Viciada e Seleção Multidesejos Aleatória com Viés. Probability Ranking seleciona um desejo usando uma lista de desejos ordenados em ordem decrescente de probabilidade de pré-condição. Loteria Viciada seleciona um desejo usando um valor numérico aleatório e intervalos numéricos – associados a desejos – proporcionais às probabilidades de suas pré-condições. Seleção Multidesejos Aleatória com Viés seleciona múltiplos desejos usando valores numéricos aleatórios e considerando as probabilidades de suas pré-condições.

Apresentamos exemplos, incluindo o agente *Vigia*, assim como experimentos envolvendo este, para mostrar como essas abordagens permitem que um agente às vezes selecione desejos cujas crenças pré-condições possuem probabilidades muito baixas.

Palavras-chave: seleção, desejos, não-determinística, seleção de desejos, incerteza, Bayes, Rede Bayesiana, agente, agente BDI, viés, viciada, loteria.

1 INTRODUCTION

Traditional software engineering does not inherently focus on systems capable of flexible autonomous actions aimed at achieving certain objectives. On the other hand, the area of autonomous agents in AI comprises entities that sense the environment in which they are situated and are capable of autonomous action in that environment. A model of autonomous agent that has been the focus of much study is the BDI agent model.

The BDI agent model (RAO; GEORGEFF, 1995) has three concepts at its center: beliefs, desires and intentions. Beliefs correspond to what the agent believes about the environment ("the world"), desires correspond to what the agent *would like* to be true, and intentions are those desires to which the agent has committed itself. In order to fulfill its desires, a BDI agent uses plans, that are adopted as intentions are created.

A traditional BDI agent does not represent uncertainty in its beliefs, and may use a closed set of ground literals to represent its beliefs. A belief is traditionally associated with a truth value, leaving no room even for *unknown* states. Uncertain knowledge supporting the notion of varying degrees of certainty/reliability – beyond the idea of a simple unknown state – can be represented through a graphical probabilistic model known as a Bayesian Network.

Bayesian Networks (PEARL, 1988) are a popular way of representing uncertain information probabilistically. They are directed acyclic graphs (DAGs), whose nodes represent event variables associated with two or more possible states. Each state has an explicit probability of occurring.

Given the lack of support for uncertainty in the BDI agent model and the representational power of Bayesian Networks, an extended agent model that integrates BDI and Bayesian Networks is provided by Fagundes and Vicari (FAGUNDES; VICARI, 2007). This BDI-based agent model no longer relies on ground literals, but rather on a Bayesian Network. This has an impact on the reasoning process; e.g., regular BDI agent processes for desire selection, action specification and plan generation are no longer appropriate.

The desire selection process is inherently susceptible to ending up choosing desires that will turn out to be impossible to satisfy under a given world state, regardless of how promising they may seem, i.e., despite any satisfactory estimate based on the Bayesian Network. On the other hand, even if an agent considers the probability of the belief preconditioning a certain desire too small, that desire cannot be guaranteed to be dispensable under uncertainty either.

This master's dissertation investigates Bayesian Networks and the workings of autonomous agents that employ them to represent uncertain beliefs, and offers approaches to desire selection alternative to the one proposed by Fagundes et al (FAGUNDES; VICARI; COELHO, 2007), so as not to disregard desires preconditioned on beliefs associated with low probabilities.

1.1 Objectives and structure

Our specific objectives are:

1. presenting what a BDI agent is;
2. presenting shortcomings of a classical BDI agent;
3. presenting Probability Theory and Bayesian Networks while seeking to avoid relying solely on formalisms;
4. presenting the union of the BDI agent paradigm and Bayesian Networks as a solution for overcoming certain limitations of the classical BDI agent;
5. presenting limitations of a previous solution and offering alternate desire selection mechanisms.

This dissertation is organized as follows: Chapter 2 focuses on the BDI agent model, Chapter 3 explains Bayesian Networks, Chapter 4 reviews Bayesian BDI agents, Chapter 5 presents approaches to Desire Selection for this particular type of agent, and Chapter 6 covers our Final Considerations for this work.

2 BDI AGENTS

Jennings (JENNINGS, 2000) defines an autonomous agent as an encapsulated computer system *situated* in an environment and capable of *flexible* autonomous action in this environment in order to achieve certain goals. The agent must adapt itself to a dynamic environment, while seeking to fulfill its goals. This chapter is concerned with Objectives 1 and 2 (the objectives are listed in Section 1.1), i.e., presenting what a BDI agent is and some of its shortcomings.

An autonomous agent model that has received attention as the subject of research and development presents beliefs, desires and intentions, being thereby called the BDI agent model (RAO; GEORGEFF, 1995). The BDI agent model is based on Bratman's theory of practical human reasoning (BRATMAN; ISRAEL; POLLACK, 1988), which is related to pursuing goals under resource constraints, establishing the means – an ordered series of logical steps – to an end. Like any agent, it can be described as having sensors and actuators, which probe and act upon the environment, respectively.

Beliefs contain a representation, internal to the agent, of environment elements considered relevant for the agent's reasoning. The state of an agent's beliefs may contain either less information that the current state of the environment (e.g., because of limited sensing ability), or more (e.g., if the agent does additional information processing on its sensing). Desires are goals that the agent would like to achieve (i.e., they can be considered an agent's *motivation* (RAO; GEORGEFF, 1995)). Intentions are those desires that the agent has committed itself to achieving. Plans are sequences of actions aimed at fulfilling intentions. Agents resist abandoning their intentions, and, should a plan fail, it is often the case that they choose to undergo replanning, despite its usually non-insignificant cost.

A BDI agent selects desires through a process that considers the current viability and the absence of conflict with existing intentions. Desires often have preconditioning beliefs that indicate whether or not they should be selected by the agent, as a matter of logical evaluation (MÓRA et al., 1999). However, changes can occur in the environment during the execution of this process. Such changes may or nor may not be meaningful with relation to which desires should be selected, if any; beliefs that precondition desires may or may not have been affected. There is no "correct" course of action: on the one hand, restarting or even rolling back the process is a waste of time if the desires could have been chosen despite the outdated (just-updated) beliefs – i.e., if the desires would have actually remained valid; on the other hand, going through with the selection of desires dependent on beliefs that in fact now contradict the environment is also an unrelished prospect. It could be argued that proceeding to planning in the latter scenario is particularly computationally wasteful, but if the frequency at which this happens is sufficiently low, it is still a justified decision. Balancing the use of these two options based on the statistical probability of each one proving appropriate and their respective measured computational costs seems to be a wise answer to this predicament.

Environment changes also occur during the obtention or the execution of a plan, possibly compromising its viability (changes affecting preconditions) and, if there is no alternate plan, that of its associated intentions. Additionally, such changes may cause higher-priority desires to become viable, which might make keeping the current intentions (i.e., maintaining commitment) be viewed as undesirable in contrast to the expected agent behavior under a given scenario. However, performing intention reconsideration poses a significant computational cost, and a change in the environment does not necessarily have an impact on an agent's intentions. Since both extremes have the potential to bring forth very objectionable results, a balance between reconsideration and commitment (i.e., a balance as to how often the agent reconsiders its intentions) must be established (KINNY; GEORGEFF, 1991).

One possible approach to intention reconsideration is to make use of reconsideration *triggers*, containing conditions that indicate that currently adopted plans have been compromised or that more important objectives (i.e., desires) have become possible (MENEGUZZI et al., 2007), thus tasking the agent designer with defining reconsideration relevance on a per-case basis. Shifting "intelligence" from an autonomous agent to its human designer is generally viewed as a practice to be cautious of, due to a possibly overwhelming number of scenarios to take into account, however, and should only be done sparingly.

2.1 Architecture

The Procedural Reasoning System (PRS) architecture (RAO; GEORGEFF, 1995) has dynamic data structures corresponding to the agent's beliefs, desires and intentions. Moreover, another present data structure is an event queue. All events (e.g., changes in the environment) are added to the queue for later processing; without some sort of control (in this case, the queue) there is a risk of computational resources being "overrun" by excessive simultaneous events.

Algorithm 1 BDI agent interpreter

```
1: procedure BDI-INTERPRETER
2:     $initializeState()$;
3:     loop
4:         $options \leftarrow optionGenerator(eventQueue)$;
5:         $selectedOptions \leftarrow deliberate(options)$;
6:         $updateIntentions(selectedOptions)$;
7:         $execute()$;
8:         $getNewExternalEvents()$;
9:         $dropSuccessfulAttitudes()$;
10:        $dropImpossibleAttitudes()$;
11:    end loop
12: end procedure
```

Pseudocode representing a PRS-type BDI agent execution cycle is presented in Algorithm 1. In this architecture, on each execution cycle what the agent initially does is collect events from the queue and obtain a list of desires (options) (Algorithm 1, Line 4);

this is important because a given desire may not be applicable given the current beliefs. Shortly thereafter, the agent selects from the options a subset (Line 5), from which intentions are generated (i.e., the agent commits itself to the chosen desires) (Line 6). The agent executes the next intention-associated action (Line 7). Then, any new external events are retrieved by the sensors and added to the event queue (Line 8). At the end of the cycle, the agent removes satisfied desires and intentions (i.e., desires that have been achieved and intentions that have been fulfilled) from their respective data structures (Line 9), as well as the impossible and inviable ones (Line 10).

Beliefs refer solely to the current time and may be represented simply as ground literals. The sets of steps for reaching possible future states are represented by *plans*, which are associated to a trigger condition and a precondition. Such plans can either by provided *a priori* in a human-devised specification or generated algorithmically (i.e., with no human intervention). Also, plans may be stored and retrieved from a *plan library*.

Chapter 3 focuses on Bayesian Networks, including Probability Theory supporting it.

3 BAYESIAN NETWORKS

3.1 Uncertainty

Uncertainty is a trait found in many domains, in the real world and in abstract worlds. This chapter deals with Objective 3 (the objectives are listed in Section 1.1), i.e., presenting Probability and Bayesian Network theory from a standpoint that is more concerned with being didactic than with employing formalisms.

Russell and Norvig (RUSSEL; NORVIG, 1994) exemplify a problem of decision under uncertainty using the following *going to the airport* scenario. A decision must be made on how soon before the flight departs the agent must leave for the airport, by car. Maybe the A_{90} plan (i.e., leaving 90 minutes in advance) will suffice, provided everything goes smoothly, that is, the traffic is normal, the car does not break down or gets involved in an accident, etc.. The alternate plan A_{120} would increase the likelihood of getting to the airport in time, but would also increase the potential wait, additionally leading to a problem of weighing priorities. Unfortunately, the data obtained from the environment via sensors do not allow for investigating the potential scenarios beforehand, so there is just no way of inferring a *correct* course of action.

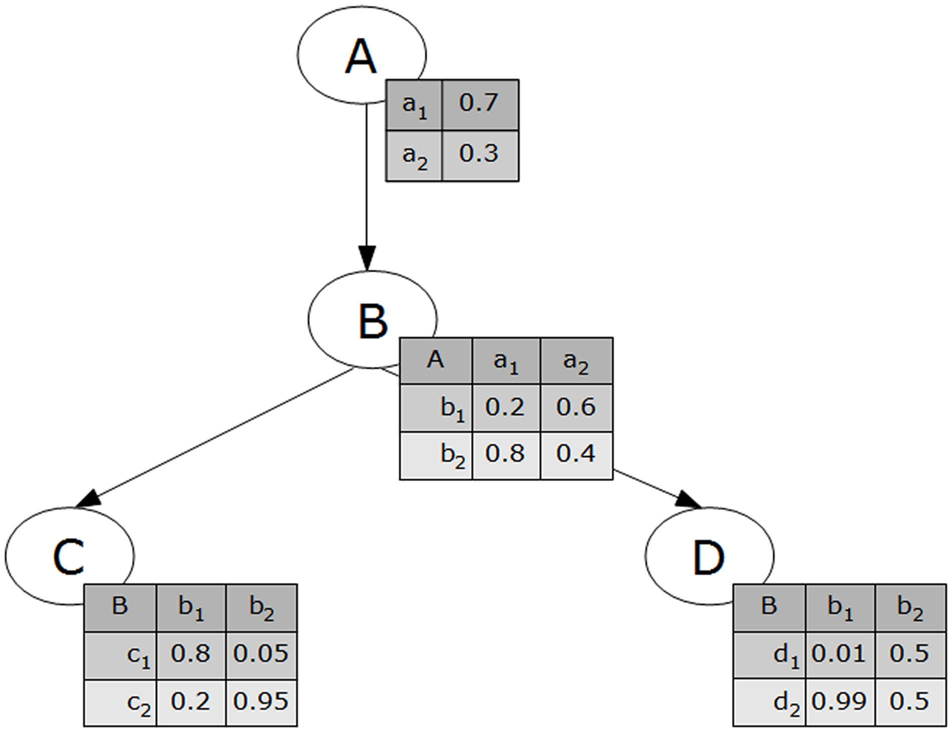
Traditional first-order logic approaches to knowledge representation are insufficient to represent certain domains where there is uncertainty in the validity of statements over time (RUSSEL; NORVIG, 1994; JENSEN; NIELSEN, 2007). Reasons for this limitation include:

- **Laziness**: the high cost of exhaustively representing all possible combinations of truth values using logic rules;

- **Theoretical ignorance**: lack of a complete theory of the domain in question (e.g., medical diagnosis);

- **Practical ignorance**: the potential impossibility or inviability of performing all necessary tests to ascertain complete truth for certain statements (e.g., (let us assume that) there is no amount of blood tests that can lead to a perfect diagnosis; all the equipment required for a given set of tests may not be available under certain time constraints).

The fact remains that people commonly reason with incomplete knowledge and make decisions based on assumptions over unknown facts. This knowledge comprises what is *known* to be true, what is *not known* and *estimates* based on relationships between elements of the world. For example, the type of reasoning humans do: “I tried to turn on my car and it didn’t work. I see two possible causes: a lack of fuel and dirty spark plugs.



a_1	0.7
a_2	0.3

A	a_1	a_2
b_1	0.2	0.6
b_2	0.8	0.4

B	b_1	b_2
c_1	0.8	0.05
c_2	0.2	0.95

B	b_1	b_2
d_1	0.01	0.5
d_2	0.99	0.5

Figure 3.1: Simple example of a Bayesian Network

Also, if one of these happened, it is unlikely the other did as well. I must look for evidence about either.” Pearl (PEARL, 1988) provides a formalism to represent partial knowledge based on the causal relationships between elements in the world, using Probability Theory to represent how knowledge about one element in the world influences the certainty about others related to it. Here, relationships between elements are represented in a network, and probabilities between related elements are calculated using Bayes’ Rule, with the resulting formalism being called a *Bayesian Network*. A Bayesian Network is a type of *causal network* that allows the specification of knowledge where parts of it are conditioned on others (e.g., cause and consequence relationships, diseases and symptoms), supporting the update of probabilities when new information (i.e., *evidence*) is obtained.

Figure 3.1 shows an example of a Bayesian Network. There are 4 event variables: A, B, C and D, each with two states. B is conditioned on A, and C and D are conditioned on B. Each variable node has an associated table with probabilities. The table for A gives us the probability of each of its states, the table for B gives us the probability of its states given each of A’s states, the table for C gives us the probability of its states given each of B’s states and the table for D gives us the probability of each of its states given each of B’s states. For instance, the probability that A is in state a_1 is 70%, and the probability that B is in state b_2, already knowing A to be in state a_1, is 80%.

In order to understand how Bayesian Networks are used to represent incomplete knowledge, it is important to understand Probability Theory. Therefore, starting next section we present the pertinent Probability Theory required to understand Bayesian Networks, present in (JENSEN; NIELSEN, 2007).

3.2 Probability Theory

Probability Theory is a foundation for mathematically probing the uncertain. It allows for estimates on possibilities. We now present some fundamentals.

3.2.1 Fundamentals

When one wishes to explore the nature of a problem that is filled with uncertainty as to its theoretically possible outcomes, one may resort to making *experiments*. An experiment, which may be viewed as a set of observations aimed at investigating causal relationships among variables, results in one of multiple possibilities, each with a probability of occurring. The set of all possible outcomes an experiment yields is referred to as the *sample space* of the experiment. A subset of the sample space is called an *event*. It is said an event A is *true* for an experiment if the outcome is an element of A.

Example 3.1 Throwing one die results in a sample space $S = \{ 1, 2, 3, 4, 5, 6 \}$, where each number has a $\frac{1}{6}$ probability of showing up. The event that throwing a six-sided die will get a number greater than 4 corresponds to the subset $\{ 5,\ 6 \} \subseteq \{ 1, 2, 3, 4, 5, 6 \}$.

Andrei Kolgomorov shows how to build the rest of Probability Theory from three axioms, named *Kolgomorov's Axioms*. The probability of each event $A \subseteq S$ must obey Kolgomorov's Axioms, presented next:

Axiom 1 $P(S) = 1$.

The event S that we will get an outcome in the sample space is certain to occur and is therefore assigned the probability 1.

Axiom 2 $\forall\ A \subseteq S$, $P(A) \geq 0$.

Any event A must have a nonnegative probability.

Axiom 3 *If* $A \subseteq S$*,* $B \subseteq S$ *and* $A\ \cap\ B = \oslash$*, then* $P(A \cup B) = P(A) + P(B)$.

If two events A and B are disjoint, then the probability of the combined event (i.e., that *at least one* of the events will be true) is the sum of the probabilities of the two individual events.

Alternatively, if A and B are not disjoint, then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Example 3.2 Consider randomly choosing a month of the year whose numeric representation is even (event A) and choosing one that has 31 days (event B). These events are not disjoint; their intersection corresponds to choosing a month in the set $\{$ *August*, *October*, *December* $\}$. Therefore the probability that a month with at least one of the specified conditions will be chosen is $\frac{6}{12} + \frac{7}{12} - \frac{3}{12} = \frac{10}{12}$.

Notation: If an event A contains only one outcome a, we write $P(a)$ rather than $P(\{a\})$.

3.2.2 Conditional probabilities

A statement about a probability is always conditioned on what else is known. For instance, in the die example, we implicitly assume that the die is fair; the probability is conditioned on this.

Notation: $P(A|B) = p$ means *"Given event B, the probability of event A is p."* If B is true, and *everything else is irrelevant for* A, then the probability of A is p.

Let A and B be subsets of S. If we know event B to take place, then all possible outcomes are elements of B, and the outcomes for which A can be true are $A \cap B$. For two events A and B, with $P(B) > 0$, the conditional probability of A given B is:

Definition 3.1: conditional probability. [1]

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Example 3.3 The conditional probability that by throwing a die (see Example 3.1) we will get number 3, given that we know that an odd number is going to show up (e.g., the die has been tampered with), is

$$P(A = \{3\}|B = \{1,3,5\}) = \frac{P(\{3\} \cap \{1,3,5\})}{P(\{1,3,5\})} = \frac{P(\{3\})}{P(\{1,3,5\})} = \frac{\frac{1}{6}}{\frac{3}{6}} = \frac{1}{3}.$$

Conditional probabilities are not restricted to just one event.

Definition 3.2: conditional probability – general version. The general definition of a conditional probability is

$$P(A|B \cap C) = \frac{P(A \cap B \cap C)}{P(B \cap C)}.$$

Assume a third event C, that we get a number that is divisible by 3. The conditional probability that we will get event A, given B and C, is

$$P(A = \{3\}|B = \{1,3,5\} \cap C = \{3,6\}) = \frac{P(\{3\} \cap \{1,3,5\} \cap \{3,6\})}{P(\{1,3,5\} \cap \{3,6\})} = \frac{P(\{3\})}{P(\{3\})} = \frac{P(3)}{P(3)} = 1,$$

since knowing B and C to be true means that the only remaining possibility is number 3.

3.2.3 Probability Calculus

If we know the probability of A given B and the probability of B, we can calculate the probability of seeing both A and B. Equation 3.1 can be rewritten so as to obtain the *fundamental rule* for Probability Calculus:

[1]The definitions in this chapter have been extracted from (JENSEN; NIELSEN, 2007).

Definition 3.3: the fundamental rule. Given events A and B,

$$P(A|B)P(B) = P(A \cap B).$$

The probability of seeing both A and B can be obtained by multiplying the probability of A given B by the probability of B. In Example 3.3, given $P(A = \{3\}|B = \{1,3,5\}) = \frac{1}{3}$ and $P(B) = \frac{1}{2}$, we can calculate $P(A \cap B) = \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{6}$, which is the same as $P(A)$. This happens because in this example $A \subseteq B$, thus $A \cap B = A$, i.e., there is no way that 3 can show up and not be an odd number.

The fundamental rule can be conditioned on another event C, as follows:

Definition 3.4: the fundamental rule – general version.

$$P(A|B \cap C)P(B|C) = P(A \cap B|C).$$

In the die example, considering event $C = \{3,6\}$ and $P(A|B \cap C) = 1$ (calculated in Example 3.3), and also

$$P(B|C) = \frac{P(B = \{1,3,5\} \cap C = \{3,6\})}{P(C = \{3,6\})} = \frac{P(\{3\})}{P(\{3,6\})} = \frac{\frac{1}{6}}{\frac{2}{6}} = \frac{1}{2},$$

we can calculate

$$P(A \cap B|C) = P(A|B \cap C)P(B|C) = 1 \cdot \frac{1}{2} = \frac{1}{2}.$$

Given that $P(A \cap B) = P(B \cap A)$ (and $P(A \cap B|C) = P(B \cap A|C)$), we get

$$\begin{aligned} P(A \cap B) &= P(B \cap A) \\ P(A|B)P(B) &= P(B|A)P(A) \\ \frac{P(A|B)P(B)}{P(B)} &= \frac{P(B|A)P(A)}{P(B)}, \end{aligned}$$ which yields *Bayes' Rule*:

Definition 3.5: Bayes' Rule. Given events A and B,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Bayes' Rule makes it possible to update beliefs about an event A, provided that we get information about another event B. Thus, $P(A)$ is usually called the *prior probability* of A, whereas $P(A|B)$ is called the *posterior probability* of A given B. $P(B|A)$ is called the *likelihood* of A given B (see Example 3.4).

There is also a general version of Bayes' Rule:

Definition 3.6: Bayes' Rule – general version. Bayes' Rule in a context C:

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}.$$

Example 3.4 (JENSEN; NIELSEN, 2007) We have two diseases a_1 and a_2, both of which can cause the symptom b. Let $P(b|a_1)$ = 0.9 and $P(b|a_2)$ = 0.3. Assume that the prior probabilities for a_1 and a_2 are the same ($P(a_1) = P(a_2)$). If b occurs, Bayes' Rule gives

$$\begin{aligned} P(a_1|b) &= \frac{P(b|a_1)P(a_1)}{P(b)} = 0.9 \cdot \frac{P(a_1)}{P(b)}; \\ P(a_2|b) &= \frac{P(b|a_2)P(a_2)}{P(b)} = 0.3 \cdot \frac{P(a_2)}{P(b)}. \end{aligned}$$

Even though the posterior probabilities cannot be calculated, as we do not know $P(b)$, we can conclude that a_1 is three times as likely as a_2 given symptom b.

3.2.4 Event independence and conditional event independence

Information on one event B may have no bearing on our belief about the occurrence of another event A. In this case, they are considered *independent*.

Definition 3.7: event independence. Any given events A and B are independent *if*

$$P(A|B) = P(A).$$

For example, when throwing two fair dice, whatever is turned up by the first die does not influence our beliefs about the probabilities relating to the second one. This notion is symmetric: if A is independent of B, so is B independent of A (i.e., $(P(A|B) = P(A)) \leftrightarrow (P(B|A) = P(B))$).

Definition 3.8: the fundamental rule for independent events. When two events are independent, the fundamental rule can be rewritten as

$$P(A \cap B) = P(A|B)P(B) = P(A) \cdot P(B).$$

This concept also exists when conditioning on multiple events. If information on event B does not change our belief about event A when we already know event C, then we say that A and B are *conditionally independent* given C.

Definition 3.9: conditional event independence. Two events are said to be conditionally independent *if*

$$P(A \cap B|C) = P(A|C) \cdot P(B|C).$$

When two events are (unconditionally) independent, it means that $C = \oslash$ (i.e., $P(A|\oslash) = P(A)$ and $P(A \cap B|\oslash) = P(A \cap B)$).

Definition 3.10: the fundamental rule for conditionally independent events. If any events A and B are independent, then

$$P(A|B \cap C) = P(A|C).$$

3.2.5 Probability Calculus for variables

Up to this point, we have considered simple events and outcomes concerning a single sample space. From now on we will be working with a collection of sample spaces, alternatively called *event variables*. A variable can be considered an experiment, and for each outcome of the experiment the variable has a corresponding *state*.

Notation: the set of states associated with a variable A is expressed as $sp(A) = (a_1, a_2, \dots, a_n)$. These states should be *mutually exclusive* and *exhaustive*. We use uppercase letters for variables and lowercase letters for states. In the die example, with a variable D representing the outcome of a die roll, $sp(D) = (1, 2, 3, 4, 5, 6)$.

For a variable A with states $a_1, \dots, a_n$, we express uncertainty about its state through a probability distribution $P(A)$ over these states:

$$P(A) = (x_1, \dots, x_n); \qquad x_i \geq 0; \qquad \sum_{i=1}^{n} x_i = x_1 + \cdots + x_n = 1,$$

where x_i is the probability of A being in state a_i. A probability distribution is called *uniform* if the probabilities of all states are equal.

Notation: the probability of A being in state a_i is denoted by $P(A = a_i)$, and denoted by $P(a_i)$ if the variable is obvious from the context.

Conditional probabilities also apply to variables: if $sp(A) = (a_1, \dots, a_n)$ and $sp(B) = (b_1, \dots, b_m)$, then $P(A|B)$ contains $n \cdot m$ conditional probabilities $P(a_i|b_j)$. This set of probabilities is usually represented in an $n \times m$ table, with one probability value for each combination of states for the variables involved. This table is called the *conditional probability table*. Additionally, Axiom 1 (Section 3.2) tells us that the probabilities over A should add up to 1 for each state of B, as shown in the next equation. Table 3.1 (JENSEN; NIELSEN, 2007) is a possible conditional probability table for $P(A|B)$.

$$\sum_{i=1}^{n} P(A = a_i | B = b_j) = 1 \text{ for each } b_j.$$

	b_1	b_2	b_3
a_1	0.4	0.3	0.6
a_2	0.6	0.7	0.4

Table 3.1: Example of a conditional probability table $P(A|B)$

The probability of seeing joint outcomes for distinct experiments can be expressed by the *joint probability* for two or more variables: for each configuration (a_i, b_j) of variables A and B, $P(A, B)$ specifies the probability of seeing both $A = a_i$ and $B = b_j$. Like $P(A|B)$, $P(A, B)$ is represented in an $n \times m$ table. This one is called the *joint probability table*. Since the sample spaces of A and B are mutually exclusive and exhaustive, all combinations of their states are mutually exclusive and exhaustive as well – as shown in the next equation, and they can be considered to constitute a sample space. Table 3.2 (JENSEN; NIELSEN, 2007) is a possible joint probability table for $P(A, B)$.

$$P(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(A = a_i, B = b_j) = 1.$$

If the sum of the probabilities of the states of the variables that constitute a given sample space is ever different than 1 as the result of a calculation, the probabilities must be *normalized.*

	b_1	b_2	b_3
a_1	0.16	0.12	0.12
a_2	0.24	0.28	0.08

Table 3.2: Example of a joint probability table $P(A, B)$

Applying the fundamental rule (Definition 3.3) to each of the $n \cdot m$ (a_i, b_j) variables, we get

$$P(a_i|b_j)P(b_j) = P(a_i, b_j).$$

If $P(B) = (0.4, 0.4, 0.2)$, Table 3.2 is the result of using the fundamental rule on Table 3.1.

Definition 3.11: the fundamental rule for variables. The fundamental rule for variables is

$$P(A, B) = P(A|B)P(B),$$

and conditioned on another variable C,

$$P(A, B|C) = P(A|B, C)P(B|C).$$

From a joint probability table $P(A, B)$, the probability distribution $P(A)$ can be calculated by considering the outcomes of B that can occur simultaneously with each state a_i of A. For each state a_i there are m mutually exclusive outcomes $(a_i, b_1), \dots, (a_i, b_m)$. Thus, by Axiom 3 (Section 3.2), we get *variable marginalization* – that of B out of $P(A, B)$.

Definition 3.12: variable marginalization. Event variable B is said to be marginalized out of $P(A, B)$ in

$$P(a_i) = \sum_{j=1}^{m} P(a_i, b_j), \text{ where } P(A) = \sum_{i=1}^{n} P(a_i).$$

Notation: $P(A) = \sum_B P(A, B)$ means that variable B is marginalized out of $P(A, B)$, the result of which is $P(A)$.

By applying this to Table 3.2, we obtain

$$P(A) = (0.16 + 0.12 + 0.12, 0.24 + 0.28 + 0.08) = (0.4, 0.6),$$

and by marginalizing out A instead of B we obtain

$$P(B) = (0.16 + 0.24, 0.12 + 0.28, 0.12 + 0.08) = (0.4, 0.4, 0.2).$$

Bayes' Rule for events can be extended to variables as well:

Definition 3.13: Bayes' Rule for variables. Given event variables A and B,

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{P(A,B)}{\sum_B P(A,B)},$$

and conditioned on another variable C,

$$P(B|A,C) = \frac{P(A|B,C)P(B|C)}{P(A|C)} = \frac{P(A,B|C)}{\sum_B P(A,B|C)}.$$

By applying Bayes' Rule using $P(A)$, $P(B)$ and $P(A|B)$ we get $P(B|A)$, shown in Table 3.3 (JENSEN; NIELSEN, 2007). Note that the probabilities over B sum to 1 for each state of A.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \begin{array}{|c|cc|}\hline & a_1 & a_2 \\ \hline b_1 & \frac{0.4\cdot0.4}{0.4} & \frac{0.6\cdot0.4}{0.6} \\ b_2 & \frac{0.3\cdot0.4}{0.4} & \frac{0.7\cdot0.4}{0.6} \\ b_3 & \frac{0.6\cdot0.2}{0.4} & \frac{0.4\cdot0.2}{0.6} \\ \hline \end{array} = \begin{array}{|c|cc|}\hline & a_1 & a_2 \\ \hline b_1 & 0.4 & 0.4 \\ b_2 & 0.3 & 0.47 \\ b_3 & 0.3 & 0.13 \\ \hline \end{array}$$

Table 3.3: Conditional probability table $P(B|A)$, given $P(A)$, $P(B)$ and $P(A|B)$. The probabilities over B sum to 1 for each state of A

Conditional independence for variables: two variables A and C are conditionally independent given variable B if

$$P(a_i|c_k, b_j) = P(a_i|b_j) \textit{ for each } a_i \in sp(A), b_j \in sp(B), \textit{ and } c_k \in sp(C).$$

Notation: $P(A|C,B) = P(A|B)$ is a more compact way of expressing conditional independence for variables. If $B = \emptyset$, A and C are said to be *marginally independent* or independent (i.e., $P(A|C) = P(A)$).

If A and C are conditionally independent given B, the fundamental rule can be simplified:

$$P(A,C|B) = P(A|B,C)P(C|B) = P(A|B)P(C|B),$$

(i.e.,

$$P(a_i, c_k|b_j) = P(a_i|b_j)P(c_k|b_j))$$

For example, by multiplying $P(A|B)$ and $P(C|B)$ – in Tables 3.1 and 3.4 (JENSEN; NIELSEN, 2007), respectively – we obtain the joint probability $P(A,C|B)$ in Table 3.5 (JENSEN; NIELSEN, 2007).

This concludes the section on Probability Calculus for variables. Having reviewed some Probability Theory required for Bayesian Networks, we proceed to sections more specifically related to the *causal network* aspect of Bayesian Networks. Section 3.3 covers the connection types found in causal networks and the property known as *d-separation*.

	b_1	b_2	b_3
c_1	0.2	0.9	0.3
c_2	0.05	0.05	0.2
c_3	0.75	0.05	0.5

Table 3.4: Conditional probability table $P(C|B)$

$$P(A, C|B) = P(A|B)P(C|B) =$$

	b_1	b_2	b_3
c_1	$(0.2 \cdot 0.4, 0.2 \cdot 0.6)$	$(0.9 \cdot 0.3, 0.9 \cdot 0.7)$	$(0.3 \cdot 0.6, 0.3 \cdot 0.4)$
c_2	$(0.05 \cdot 0.4, 0.05 \cdot 0.6)$	$(0.05 \cdot 0.3, 0.05 \cdot 0.7)$	$(0.2 \cdot 0.6, 0.2 \cdot 0.4)$
c_3	$(0.75 \cdot 0.4, 0.75 \cdot 0.6)$	$(0.05 \cdot 0.3, 0.05 \cdot 0.7)$	$(0.5 \cdot 0.6, 0.5 \cdot 0.4)$

$=$

	b_1	b_2	b_3
c_1	$(0.08, 0.12)$	$(0.27, 0.63)$	$(0.18, 0.12)$
c_2	$(0.02, 0.03)$	$(0.015, 0.035)$	$(0.12, 0.08)$
c_3	$(0.3, 0.45)$	$(0.015, 0.035)$	$(0.3, 0.2)$

Table 3.5: Joint probability table $P(A, C|B)$, given that A and C are conditionally independent given B

3.3 Connection types and d-separation

Probability tables grow exponentially with the number of variables. The main advantage of Bayesian Networks over exhaustive tables – full joint probability tables – is that they do not require that all the variables in the network be taken into account simultaneously; there can be variables independent of each other and these do not increase the computational cost as though they had an impact on the associated probabilities. The property in Bayesian Networks that tells us that a variable is independent of another is called *d-separation*.

There may be evidence that a given variable is in a certain state. When this happens, it is said that such a variable is *instantiated*. This kind of evidence is called *hard evidence*. Conversely, if a statement about a variable state is made based on dependencies rather than explicit knowledge, it is said that there is *soft evidence* about that variable.

The topology of causal networks comprises instances of three types of connection: serial, diverging and converging. Each connection type accounts for a specific reasoning as to whether variables are *d-separated* or *d-connected* (what we call variables that are not d-separated) [2].

3.3.1 Serial

In a serial connection, if we have no hard evidence about a variable, evidence about its parent/child passes through it, affecting our beliefs about it and about its uninstantiated child/parent. Imagine three variables, A, B and C, where A is connected to B and B is connected to C, as in Figure 3.2 (JENSEN; NIELSEN, 2007).

[2]"d" is for "directed graph".

A influences B and B influences C. Thus, evidence about A has an impact on the belief about the certainty of B, which in turn influences the belief about the certainty of C. Conversely, if C receives evidence, it affects B, which in turn affects A. These statements are true, however, only if there is no hard evidence about B (i.e., if the state of B is known, it "blocks communication" between A and C). If B is instantiated, A and C are said to be *d-separated*; otherwise, they are *d-connected*.

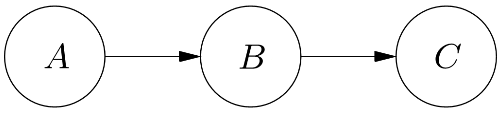


Figure 3.2: Serial connection. If B is instantiated, A and C are *d-separated*.

Example 3.5: Rainfall, Water level, Flooding (JENSEN; NIELSEN, 2007). The example in Figure 3.3 (JENSEN; NIELSEN, 2007) is a causal model consisting of a serial connection among Rainfall (no, light, medium, heavy), Water level (low, medium, high) and Flooding (yes, no)[3]. If there is heavy rain, the odds of a high water level increase, which in turn increases the odds of flooding. Conversely, knowing that there has been a flooding increases our belief that the water level is high, which in turn increases our belief that there has been a heavy rain. However, if we already know the water level, neither rainfall nor flooding affects our belief about the other (i.e., in this case they are *d-separated*).

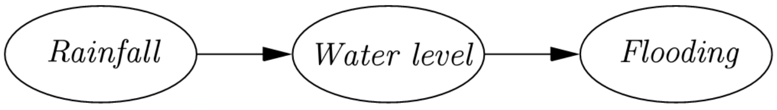


Figure 3.3: Causal model for rainfall, water level and flooding using a serial connection.

3.3.2 Diverging

In a diverging connection, one variable is connected to two or more variables. If we have no hard evidence concerning the parent, evidence about one of its children affects our beliefs about the other – uninstantiated – children. Imagine five variables, A, B, C, D and E, where A is connected to all the others, as in Figure 3.4 (JENSEN; NIELSEN, 2007). B, C, D and E can transmit evidence among themselves through A, unless A is instantiated, blocking communication between its children. $B, \ldots, E$ are said to be *d-separated* if A is instantiated and *d-connected* if not.

Example 3.6: Sex, Hair length, Stature (JENSEN; NIELSEN, 2007). The example in Figure 3.5 (JENSEN; NIELSEN, 2007) is a causal model consisting of a diverging connection involving Sex (male, female), Hair length (short, long) and Stature ($<$ 170 cm, $\geq$ 170 cm). If a person's sex is not known, seeing his/her stature is relevant to our belief as to the sex, which in turn is relevant as to the hair length. However, if we already know a person's sex, knowing the stature is no help for estimating the hair length, (i.e., stature and hair length are then *d-separated*).

[3]**Notation:** <variable name> (<variable states separated by comma>)

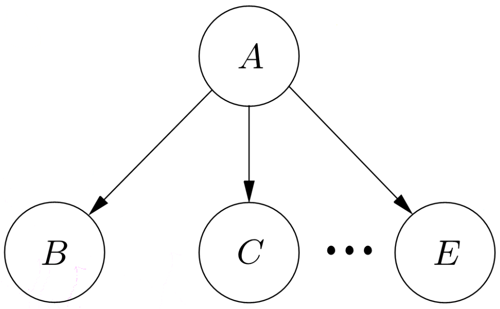


Figure 3.4: Diverging connection. If A is instantiated, its children are *d-separated.*

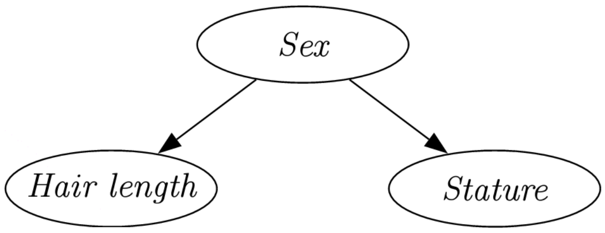


Figure 3.5: Causal model for sex, hair length and stature using a diverging connection.

3.3.3 Converging

In a converging connection, two or more variables, unconnected among themselves, are connected to one other (their child), which may have descendants. If we have no hard evidence about the child or one of its descendants, evidence about a parent does not influence our beliefs about the other(s). Imagine five variables, A, B, C, D and E, where B, C, D and E are connected to A, but not to each other, as in Figure 3.6 (JENSEN; NIELSEN, 2007). Figure 3.7 (JENSEN; NIELSEN, 2007) shows two other causal model samples that use a converging connection. Only if A or one of its descendants is instantiated can A's parents – B, C, ..., E in Figure 3.6 and B, C in Figure 3.7 – transmit evidence among themselves through A. If A and its descendants are not instantiated, A's parents are said to be *d-separated.* Otherwise, they are *d-connected.*

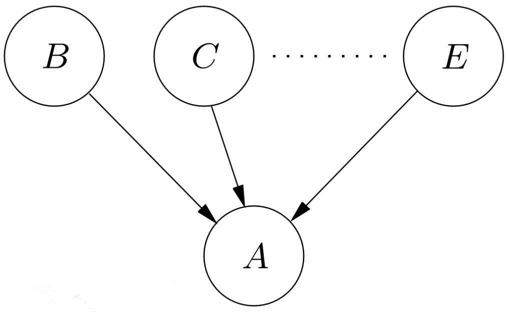


Figure 3.6: Converging connection. If A is not instantiated, and neither is any of its descendants, its parents are *d-separated.*

Example 3.7: Fuel, Clean Spark Plugs, Start (JENSEN; NIELSEN, 2007). Let's look at a problem: if the car cannot start, the potential causes are dirty spark plugs and an empty fuel tank . If we find out that there is fuel in the tank, our belief that the spark plugs are dirty will increase, since it is the other possible explanation. If we find out that there

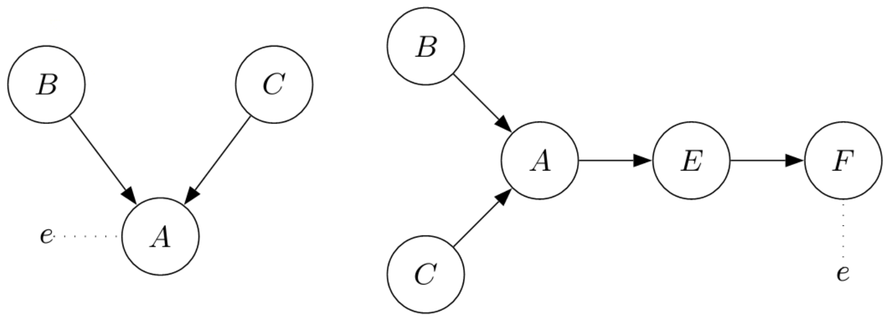


Figure 3.7: Converging connections. In both graphs, given the hard evidence denoted by $e \cdots$, B and C are *d-connected*.

is no fuel in the tank, then, since it is less likely that there is a problem with *both* the fuel and the spark plugs, our belief that the spark plugs are dirty will decrease (i.e., fuel and spark plugs will be *d-connected*). Conversely, without the information on whether the car can or cannot start, evidence about the fuel or the spark plugs will not matter to the other (i.e., they will be independent possible issues; *d-separated*). A causal model for this is shown in Figure 3.8 (JENSEN; NIELSEN, 2007).

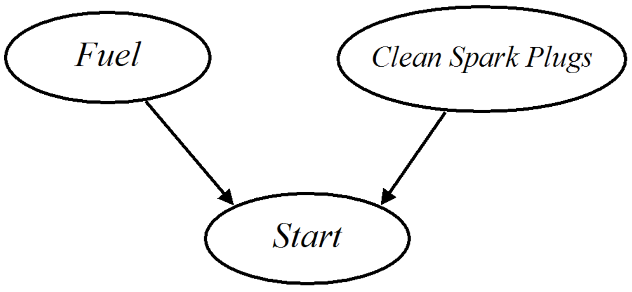


Figure 3.8: Causal model for the “Car Cannot Start” problem using a converging connection.

Definition 3.14: d-separation. *Two variables A and B are d-separated if for all paths between A and B there is an intermediate variable V, other than A and B, such that either*

- the connection is serial or diverging and V is instantiated

or

- the connection is converging, and neither V nor any of V's descendants have received evidence.

We need a computationally feasible way of determining whether any two variables are d-separated given a set of variables about which we have hard evidence. Section 3.4 explains how this can be achieved.

3.4 Moralization

Suppose we want to find out whether variables A and B are d-separated given hard evidence on a set of variables C. Consider the causal network in Figure 3.9 (JENSEN; NIELSEN, 2007). The usual way of doing this is to determine whether all paths connecting A and B are d-separating. In order to avoid the burden of having to check all paths connecting the corresponding nodes so we can determine d-separation for the variables, we execute the procedure known as *moralization*.

The first step is to create the so-called *ancestral graph*, consisting of A, B and C together with all nodes from which there is a directed path to one of them. Next, we insert an undirected link between each pair of nodes with a common child and make all links undirected. The resulting graph is known as the *moral graph* for that causal network. The moral graph, rather than the causal network, can now be used to check whether A and B are d-separated given C: if all paths connecting A and B intersect at least one variable in set C, then A and B are d-separated given C.

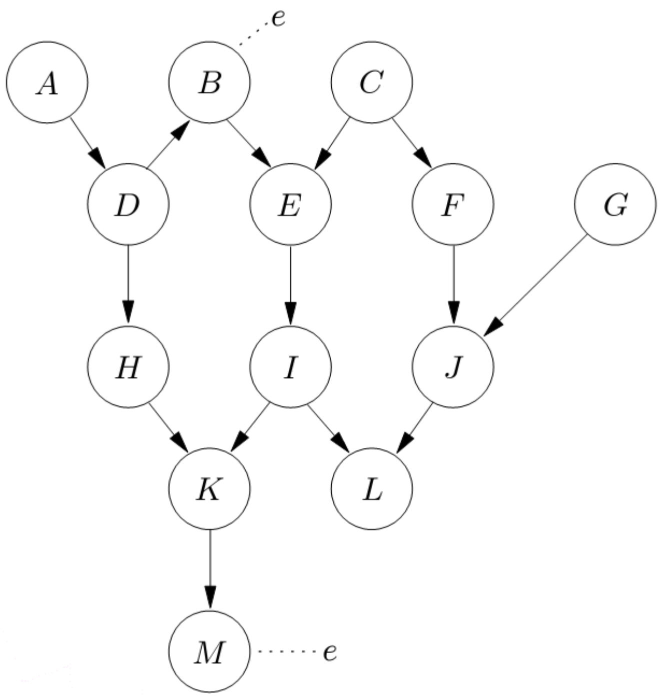


Figure 3.9: Example of a causal network to be moralized.

To test whether A is d-separated from F given evidence on B and M in Figure 3.9 (JENSEN; NIELSEN, 2007), we first construct the ancestral graph for A,B,F,M (Figure 3.10(a) (JENSEN; NIELSEN, 2007)). Next we add an undirected link between pairs of nodes with a common child and then the direction is dropped on all links (Figure 3.10(b) (JENSEN; NIELSEN, 2007)). In the resulting graph we have that the path A-D-H-K-I-E-C-F does not intersect B and M, hence A and F are d-connected given B and M.

3.5 Model structure

Bayesian Networks are a specialization of causal networks consisting of directed acyclic graphs whose nodes are each associated with a variable. Each variable in a Bayesian Network has a set of mutually exclusive states (i.e., a corresponding state space). The variables in a Bayesian Network may or may not be conditioned on others. Probabilistic information is provided in *conditional probability tables* or, when the nodes do not have parents, *unconditional probability tables*.

Definition 3.15: Bayesian Network. *A Bayesian Network consists of the following:*

- *A set of variables and a set of directed edges between variables.*

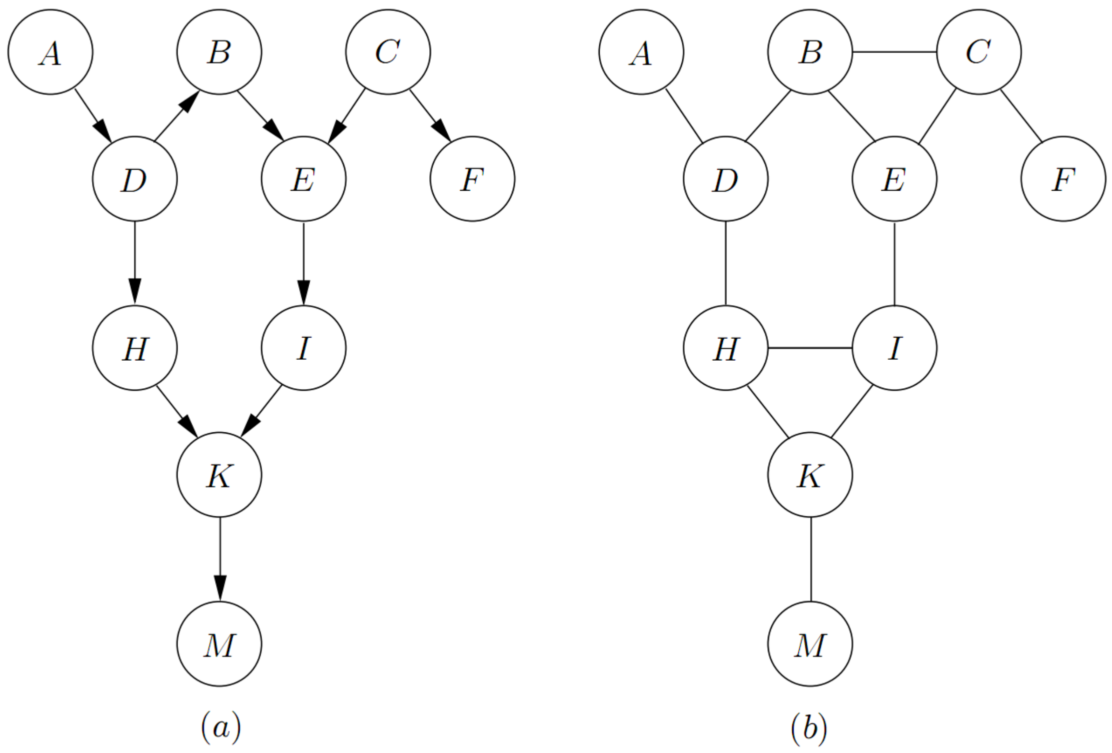


Figure 3.10: Moralization example in two steps.

- *Each variable has a set of mutually exclusive states.*

- *The variables together with the directed edges form a directed acyclic graph (DAG); a graph is acyclic if there is no directed path $A_1 \longrightarrow \ldots \longrightarrow A_n$ so that $A_1 = A_n$.*

- *To each variable A with parents $B_1, \ldots, B_n$ a conditional probability table $P(A| B_1, \ldots, B_n)$ is assigned. To each variable A with no parents, an unconditional probability table $P(A)$ (i.e., a table containing the prior probabilities of A) is assigned.*

Bayesian Networks do not require that the links represent causal impact (i.e., when building the structure of a Bayesian Network model, we do not *have* to make the links go in a causal direction). However, this also means that we should check the model's d-separation properties so they reflect our perception of the world's conditional independence relationships.

Given the causal model for the car start problem (Example 3.7) shown in Figure 3.11 (JENSEN; NIELSEN, 2007), where Fu refers to *Fuel?*, SP to *Clean Spark Plugs?*, St to *Start?* and FM to *Fuel Meter Standing?*, we need the probabilities $P(Fu)$, $P(SP)$, $P(St|Fu, SP)$ and $P(FM|Fu)$. Let $P(Fu) = (0.98, 0.02)$ and $P(SP) = (0.96, 0.04)$. See also Table 3.6 (JENSEN; NIELSEN, 2007) and note that $P(FM|Fu)$ corresponds to a possible malfunction in the fuel meter and $P(St|Fu, SP)$ admits the possibility of a cause other than lack of fuel and dirty spark plugs ($P(St = no|Fu = yes, SP = yes) = 0.01$).

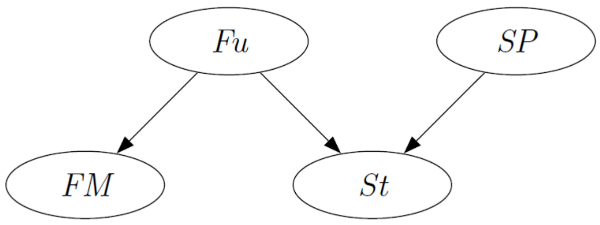


Figure 3.11: Causal model for the car start problem.

$P(FM|Fu) =$

	$Fu = yes$	$Fu = no$
$FM = full$	0.39	0.001
$FM = \frac{1}{2}$	0.6	0.001
$FM = empty$	0.01	0.998

$P(St|Fu, Sp) =$

	$Fu = yes$	$Fu = no$
$SP = yes$	$(0.99, 0.01)$	$(0, 1)$
$SP = no$	$(0.01, 0.99)$	$(0, 1)$

Table 3.6: Conditional probabilities for the car start problem represented in Figure 3.11.

3.6 The chain rule for Bayesian Networks

Let $\mathcal{U} = A_1, \ldots, A_n$ be a universe of variables. If we have access to the joint probability table $P(\mathcal{U}) = P(A_1, \ldots, A_n)$, then we can also calculate $P(A_i)$ as well as $P(A_i|e)$, where e is evidence about some of the variables in the Bayesian Network. However, $P(\mathcal{U})$ grows exponentially with the number of variables, and $\mathcal{U}$ does not need to be very large before the table becomes intractably large. Therefore, we look for a more compact representation of $P(\mathcal{U})$ (i.e., a way of storing information from which $P(\mathcal{U})$ can be calculated if needed).

Let BN be a Bayesian Network over $\mathcal{U}$, and let $P(\mathcal{U})$ be a probability distribution reflecting the properties specified by BN: *(i)* the conditional probabilities for a variable given its parents in $P(\mathcal{U})$ must be as specified in BN, and *(ii)* if the variables A and B are d-separated in BN given the set C, then A and B are independent given C in $P(\mathcal{U})$.

For probability distributions over sets of variables, we have an equation called *the chain rule*. For Bayesian Networks this equation has a special form. First we state the general chain rule:

Proposition 3.1: The general chain rule (JENSEN; NIELSEN, 2007). *Let $\mathcal{U} = A_1, \ldots, A_n$ be a set of variables. Then for any probability distribution $P(\mathcal{U})$ we have*

$$P(\mathcal{U}) = P(A_n|A_1, \ldots, A_{n-1})P(A_{n-1}|A_1, \ldots, A_{n-2}) \ldots P(A_2|A_1)P(A_1).$$

Theorem 3.1: The chain rule for Bayesian Networks (JENSEN; NIELSEN, 2007). Let BN be a Bayesian Network over $\mathcal{U} = A_1, \ldots, A_n$. Then BN specifies a unique joint probability distribution $P(\mathcal{U})$ given by the product of all conditional probability tables specified in BN:

$$P(\mathcal{U}) = \prod_{i=1}^{n} P(A_i|pa(A_i)),$$

where $pa(A_i)$ are the parents of A_i in BN, and $P(\mathcal{U})$ reflects the properties of BN.

The chain rule yields that a Bayesian Network is a compact representation of a joint probability distribution. Example 3.8 illustrates how to exploit that for reasoning under uncertainty.

Example 3.8: The Car Start problem revisited (JENSEN; NIELSEN, 2007). Here we apply the rules of Probability Calculus to the "car start problem" to perform reasoning. We use the joint probability table, which is calculated from the chain rule for Bayesian Networks,

$$P(Fu, FM, SP, St) = P(Fu)P(SP)P(FM|Fu)P(St|Fu, SP).$$

The calculations are given in Tables 3.7 and 3.8 (JENSEN; NIELSEN, 2007), where the numeric values (x, y) represent $P(Fu = yes, Fu = no)$.

	$FM = full$	$FM = \frac{1}{2}$	$FM = empty$
$SP = yes$	$(0.98 \cdot 0.96 \cdot 0.39 \cdot 0.99, 0.02 \cdot 0.96 \cdot 0.001 \cdot 0)$	$(0.98 \cdot 0.96 \cdot 0.6 \cdot 0.99, 0.02 \cdot 0.96 \cdot 0.001 \cdot 0)$	$(0.98 \cdot 0.96 \cdot 0.01 \cdot 0.99, 0.02 \cdot 0.96 \cdot 0.998 \cdot 0)$
$SP = no$	$(0.98 \cdot 0.04 \cdot 0.39 \cdot 0.01, 0.02 \cdot 0.04 \cdot 0.001 \cdot 0)$	$(0.98 \cdot 0.04 \cdot 0.6 \cdot 0.01, 0.02 \cdot 0.04 \cdot 0.001 \cdot 0)$	$(0.98 \cdot 0.04 \cdot 0.01 \cdot 0.01, 0.02 \cdot 0.04 \cdot 0.998 \cdot 0)$

=

	$FM = full$	$FM = \frac{1}{2}$	$FM = empty$
$SP = yes$	$(0.363, 0)$	$(0.559, 0)$	$(0.0093, 0)$
$SP = no$	$(0.00015, 0)$	$(0.00024, 0)$	$(3.9 \cdot 10^{-6}, 0)$

Table 3.7: The joint probability table for $P(Fu, FM, SP, St = yes)$.

	$FM = full$	$FM = \frac{1}{2}$	$FM = empty$
$SP = yes$	$(0.98 \cdot 0.96 \cdot 0.39 \cdot 0.01, 0.02 \cdot 0.96 \cdot 0.001 \cdot 1)$	$(0.98 \cdot 0.96 \cdot 0.6 \cdot 0.01, 0.02 \cdot 0.96 \cdot 0.001 \cdot 1)$	$(0.98 \cdot 0.96 \cdot 0.01 \cdot 0.01, 0.02 \cdot 0.96 \cdot 0.998 \cdot 1)$
$SP = no$	$(0.98 \cdot 0.04 \cdot 0.39 \cdot 0.99, 0.02 \cdot 0.04 \cdot 0.001 \cdot 1)$	$(0.98 \cdot 0.04 \cdot 0.6 \cdot 0.99, 0.02 \cdot 0.04 \cdot 0.001 \cdot 1)$	$(0.98 \cdot 0.04 \cdot 0.01 \cdot 0.99, 0.02 \cdot 0.04 \cdot 0.998 \cdot 1)$

=

	$FM = full$	$FM = \frac{1}{2}$	$FM = empty$
$SP = yes$	$(0.00367, 1.9 \cdot 10^{-5})$	$(0.00564, 1.9 \cdot 10^{-5})$	$(9.4 \cdot 10^{-5}, 0.0192)$
$SP = no$	$(0.01514, 8 \cdot 10^{-7})$	$(0.0233, 8 \cdot 10^{-7})$	$(0.000388, 0.000798)$

Table 3.8: The joint probability table for $P(Fu, FM, SP, St = no)$.

The evidence $St = no$ tells us that we are in the context of Table 3.8. By marginalizing FM and Fu out of Table 3.8, summing each row, we get

$$P(SP, St = no) = (0.02864, 0.03965).$$

We get the conditional probability $P(SP|St = no)$ by dividing $P(SP, St = no)$ by $P(St = no)$. $P(St = no) = P(SP = yes, St = no) + P(SP = no, St = no) =$

$0.02864 + 0.03965 = 0.06829$, so

$$P(SP|St = no) = \left(\frac{0.02864}{0.06829}, \frac{0.03965}{0.06829}\right) = (0.42, 0.58).$$

Similarly, we get $P(Fu|St = no) = (0.71, 0.29)$. Suppose that we now get hard evidence that $FM = \frac{1}{2}$. This means we are now limited to $P(Fu, SP, St = no, FM = \frac{1}{2})$, shown in Table 3.9 (JENSEN; NIELSEN, 2007).

	$Fu = yes$	$Fu = no$
$SP = yes$	0.00564	$1.9 \cdot 10^{-5}$
$SP = no$	0.0233	$8 \cdot 10^{-7}$

Table 3.9: $P(Fu, SP, St = no, FM = \frac{1}{2})$

By marginalizing SP out and normalizing, we get $P(Fu|St = no, FM = \frac{1}{2}) = (0.999, 0.001)$, and by marginalizing Fu out and normalizing we get $P(SP|St = no, FM = \frac{1}{2}) = (0.196, 0.804)$. The probability of $SP = yes$ increased by observing $FM = \frac{1}{2}$.

3.7 Inserting evidence

Assume $P(A) = (x_1, \ldots, x_n)$ and that hard evidence e indicating that only the i and j states are now possible has been obtained. This means that $P(A, e) = (0, \ldots, 0, x_i, 0, \ldots, 0, x_j, 0, \ldots, 0)$. This event is called a *finding*. $P(e)$ can be obtained by marginalizing A out of $P(A, e)$. Additionally, $P(A, e)$ is the result of multiplying $P(A)$ by $(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0)$.

Definition 3.16: finding. *Let A be a variable with n states. A finding on A is an n-dimensional table of zeros and ones.*

Assume a joint probability table $P(\mathcal{U})$ and a finding e. The joint probability table $P(\mathcal{U}, e)$ is the table obtained from $P(\mathcal{U})$ by replacing all entries with A not in states i or j with the value zero and leaving all other entries unchanged. This is equivalent to

$$P(\mathcal{U}, e) = P(\mathcal{U}) \cdot e$$

Note that $P(e) = \sum_U P(\mathcal{U}, e) = \sum_U (P(\mathcal{U}) \cdot e)$. Using the chain rule for Bayesian Networks, we have the following theorem:

Theorem 3.2 (JENSEN; NIELSEN, 2007) *Let BN be a Bayesian Network over the universe $\mathcal{U}$, and let $e_1, \ldots, e_m$ be findings. Then*

$$P(\mathcal{U}, e) = \prod_{A \in \mathcal{U}} P(A|pa(A)) \cdot \prod_{i=1}^{m} e_i,$$

and for $A \in \mathcal{U}$ we have

$$P(A|e) = \frac{\sum_{\mathcal{U} \setminus \{A\}} P(\mathcal{U}, e)}{P(e)}.$$

3.8 Calculating probabilities in practice

Since calculating the full joint probability table $P(\mathcal{U})$ can easily become infeasible in practice as $\mathcal{U}$ grows, an approach that does not rely on that must be taken.

Consider the Bayesian Network in Figure 3.12 and assume that all variables have ten states. Given the evidence $e = \{D = d, F = f\}$, we wish to calculate $P(A|e)$.

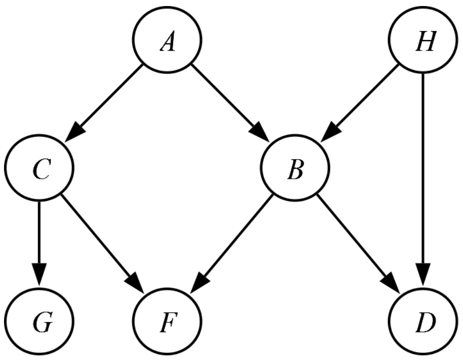


Figure 3.12: A Bayesian Network

From the chain rule we have

$$\begin{aligned} P(\mathcal{U}, e) &= P(A, B, C, d, f, G, H) \\ &= P(A)P(H)P(B|A, H)P(C|A)P(d|B, H)P(f|B, C)P(G|C), \end{aligned}$$

where for example $P(d|B, H)$ denotes the table over B and H resulting from considering only the state d of D (i.e., the conditional probability table instantiated to $D = d$). There is no need to calculate the full table $P(\mathcal{U})$ with 10^7 entries.

To calculate $P(A, e)$, we marginalize B, C, G and H out of $P(A, B, C, d, f, G, H)$. The order of the variables in the marginalization has no impact on the result. Let's start with G:

$$\begin{aligned} P(A, B, C, d, f, H) &= \sum_G P(A, B, C, d, f, G, H) \\ &= \sum_G P(A)P(H)P(B|A, H)P(C|A)P(d|B, H)P(f|B, C)P(G|C) \\ &= P(A)P(H)P(B|A, H)P(C|A)P(d|B, H)P(f|B, C)\sum_G P(G|C). \end{aligned}$$

For each state c of C we have $\sum_G P(G|c) = 1$, therefore

$$\begin{aligned} P(A, B, C, d, f, H) &= \sum_G P(A, B, C, d, f, G, H) \\ &= P(A)P(H)P(B|A, H)P(C|A)P(d|B, H)P(f|B, C). \end{aligned}$$

Now we pick H for marginalization.

$$\begin{aligned} P(A, B, C, d, f) &= \sum_H P(A, B, C, d, f, H) \\ &= P(A)P(C|A)P(f|B, C)\sum_H P(H)P(B|A, H)P(d|B, H). \end{aligned}$$

We multiply the three tables $P(H)$, $P(B|A, H)$, and $P(d|B, H)$, and we marginalize H out of the product. The result is a table $\mathcal{T}(d, B, A)$, and we have

$$P(A, B, C, d, f) = P(A)P(C|A)P(f|B, C)\mathcal{T}(d, B, A).$$

Now we calculate this product and marginalize B and C out of it.

We never work with a table of more than three variables (the table produced by multiplying $P(H)$, $P(B|A, H)$, and $P(d|B, H)$) compared to the five variables in $P(A, B, C, d, f, G, H)$. This method is called *variable elimination* and can be described as: we start with a set $\mathcal{T}$, and whenever we wish to marginalize a variable $\mathcal{X}$ in their domains, calculate the product of them, marginalize $\mathcal{X}$ out of it, and place the resulting table in $\mathcal{T}$.

Next is a summary of Bayesian Networks.

3.9 Summary

The following is a summary of formulae related to Bayesian Networks, for easy reference:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$
$$P(A|B \cap C) = \frac{P(A \cap B \cap C)}{P(B \cap C)}$$

The fundamental rule

$$P(A|B)P(B) = P(A \cap B)$$
$$P(A|B \cap C) = P(A \cap B|C)$$

Bayes' Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$
$$P(A|B, C) = \frac{P(B|A, C)P(A|C)}{P(B|C)}$$

$P(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(A = a_i, B = b_j) = 1.$

The fundamental rule for variables

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{P(A,B)}{\sum_B P(A,B)}$$
$$P(B|A,C) = \frac{P(A|B,C)P(B|C)}{P(A|C)} = \frac{P(A,B|C)}{\sum_B P(A,B|C)}$$

If $P(A|C,B) = P(A,B)$ *then* A *and* C *are conditionally independent given* B .

$P(A,B,C) = P(A|B,C)P(B,C)$

Next, we will talk about inference.

3.10 Bayesian Network inference

Asking questions about the knowledge represented in a Bayesian Network may ultimately mean inquiring about the state of the variables that constitute said network. Obtaining from a Bayesian Network probabilistic values you would from a full joint probability table (FJPT) is called performing *inference*. You can compute a query without explicitly creating the FJPT, but its time complexity is still at least $O(2^n)$ for n Boolean variables (2 comes from the two possible states of a Boolean variable).

JavaBayes (COZMAN, 2000) and JAmplia (MACHADO; VICARI, 2006) contain implementations of exact Bayesian Network inference. JavaBayes employs an approach of its own based on variable elimination, while JAmplia relies on multiple graph-theory algorithms applied in a certain order that ultimately produce a *JunctionTree*. However, neither provides an actual API.

Finally, at the very end of this work we found SMILE (Structural Modeling, Inference, and Learning Engine) (PITTSBURGH, 2013), a free engine that provides an API that allows you to create Bayesian Networks and perform Bayesian inference on them. SMILE is not open-source, however.

Chapter 4 focuses on Bayesian BDI Agents.

4 BAYESIAN BDI AGENTS

4.1 Introduction

Artificial Intelligence often comes across situations where it easily becomes infeasible to deal with every contingency. Although there are scenarios for which uncertainty does not have to be modelled, these can be considered a subset of real-world scenarios. The very fact that uncertainty permeates reality serves as evidence of the limited applicability of an autonomous agent model that does not cope with the unknown. In order to address this need, Fagundes and Vicari (FAGUNDES; VICARI, 2007) provide an effort to integrate the BDI agent model and Bayesian Networks. Additionally, Kieling and Vicari (KIELING; VICARI, 2011) integrate Bayesian Networks into an implementation of the Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007) AgentSpeak(L) (RAO; GEORGEFF, 1995) interpreter; in their work, desires are not given an explicit representation.[1] Finally, Carrera and Iglesias (CARRERA; IGLESIAS, 2012) focus on the process of updating beliefs within a Bayesian BDI agent.

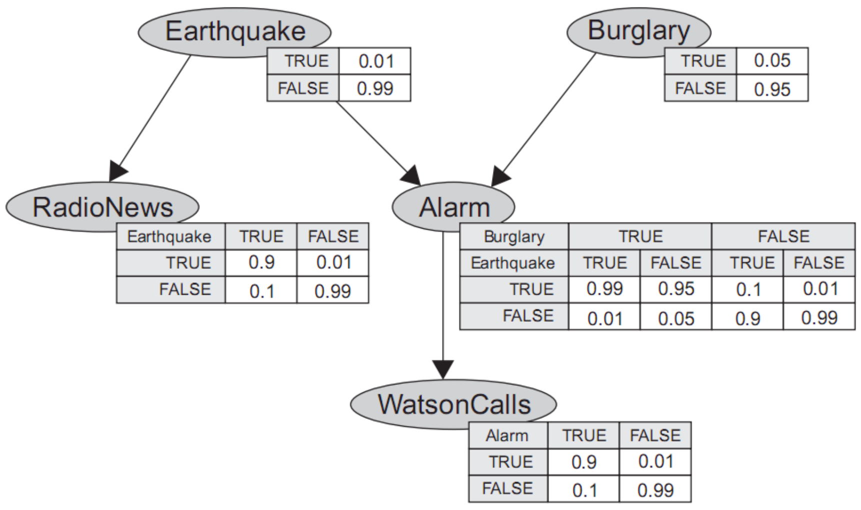
This chapter is concerned with Objective 4 (the objectives are listed in Section 1.1), i.e., presenting Bayesian BDI agents.

4.2 Running example

Let us consider the *Burglary or Earthquake* problem (PEARL, 1988), which we use later. Mr. Holmes is working in his office and then receives a call from Mr. Watson, who tells him that his alarm has gone off. Believing a burglar to have broken into his house, Mr. Holmes rushes home. On his way home, he turns on the radio and listens to news of a small earthquake having occurred in the area. Aware that earthquakes tend to fire off burglar alarms, he returns to work. Figure 4.1 (FAGUNDES; VICARI, 2007) is an example of a Bayesian Network that illustrates this problem.

In this Bayesian Network, it is stated that the probability of an $Earthquake$ taking place is 1%, and that the probability of a $Burglary$ taking place is 5%. If it is observed that an $Earthquake$ has occurred, it is stated that there is a 90% probability of Mr. Holmes hearing $RadioNews$ of it, and a 1% probability of him hearing this news while there not being an $Earthquake$ (i.e., false news). Moreover, if an $Earthquake$ takes place and a $Burglary$ does happen, the probability of the $Alarm$ in Mr. Holmes' home firing off is 99%, while if it is known that only the $Earthquake$ has occurred, the probability of the $Alarm$ being activated is a mere 1%. A $Burglary$ happening without the $Earthquake$ yields a 95% probability of the $Alarm$ firing off. If neither an

[1]As is also the case with Jason itself (MENEGUZZI, 2009).



Earthquake	
TRUE	0.01
FALSE	0.99

Burglary	
TRUE	0.05
FALSE	0.95

Earthquake	TRUE	FALSE
TRUE	0.9	0.01
FALSE	0.1	0.99

Burglary	TRUE		FALSE	
Earthquake	TRUE	FALSE	TRUE	FALSE
TRUE	0.99	0.95	0.1	0.01
FALSE	0.01	0.05	0.9	0.99

Alarm	TRUE	FALSE
TRUE	0.9	0.01
FALSE	0.1	0.99

Figure 4.1: *Burglary or Earthquake* Bayesian Network

$Earthquake$ nor a $Burglary$ is observed, it is stated that the probability of the $Alarm$ going off is 1%.

4.3 Connection types and variable dependencies

Figure 4.2 (FAGUNDES; VICARI, 2007) presents instances of the three connection types present in the Bayesian Network in Figure 4.1 (Section 3.3 deals with the three connection types found in Bayesian Networks and the *d-separation* criterion that indicates whether or not evidence may be transmitted between variables given certain evidence). If the state of the $Alarm$ variable is known (i.e., there is *evidence* about it), then evidence on $Burglary$ will not affect $WatsonCalls$ (i.e., $Burglary$ and $WatsonCalls$ are *d-separated*). Otherwise, beliefs about $Burglary$ influence beliefs about $WatsonCalls$. If it is observed that the $Alarm$ has gone off, the beliefs about $WatsonCalls$ are updated. Conversely, one may update the beliefs on $Alarm$ if the state of $WatsonCalls$ is observed (abductive reasoning). Furthermore, if the $Alarm$ is known to have fired, it means that chances of there having been an $Earthquake$ are affected (*d-connected* through a serial connection), as are those of hearing $RadioNews$ about it ($Alarm$ and $RadioNews$ *d-connected* through a diverging connection from (uninstantiated) $Earthquake$). It also means that $Earthquake$ and $Burglary$ affect each other: if one happened, the chances of the other having happened as well are decreased (*d-connected* through a converging connection to (instantiated) $Alarm$). If the state of the $Alarm$ is not known, $Earthquake$ and $Burglary$ have no impact on each other.

4.4 Modus operandi

We outline a generic reasoning cycle for a Bayesian BDI agent situated in an uncertain environment, in Algorithm 2. First, the agent updates its belief base, i.e., its Bayesian Network, according what it has perceived from the environment (Line 2). Second, the agent evaluates if each desire has been satisfied, removing it from the list of desires if so (Line 3). The agent then proceeds to evaluate its possible desires (Line 4) – the desire selection process itself is the subject of Chapter 5. If a desire has been selected,

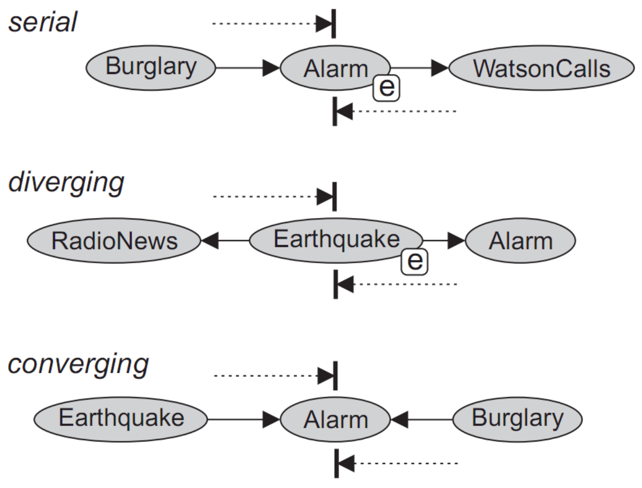


Figure 4.2: *Burglary or Earthquake* – connection types

Algorithm 2 Generic Reasoning Cycle for Bayesian BDI Agents

```
 1: procedure GENERIC BDI REASONING CYCLE
 2:     update beliefs based on percept
 3:     evaluate desire satisfaction and remove fulfilled desires
 4:     evaluate and possibly choose desires
 5:     seek plans that might satisfy the chosen desires
 6:     for each chosen desire, if an applicable plan has been found, create an intention
        and associate it with the desire and the plan, which is therefore adopted
 7:     if a plan was not found, mark the desire as unsatisfiable at this time
 8:     if adopted plan failed, either seek another plan or remove the intention (subject to
        commitment policy)
 9:     if adopted plan succeeded, remove the intention
10: end procedure
```

the agent must attempt to bring upon itself the intention to satisfy it, i.e., commit to the desire. With this in mind, it seeks plans that seem capable of satisfying it through a certain sequence of actions (Line 5). We say "seem" because a plan might turn out to be ineffective, despite a successful precondition validation, as the preconditions themselves may have been uncertain (they were, unless there was hard evidence on them), *and* there may have been unexpected changes in the environment (e.g., caused by other agents). Having obtained a set of applicable plans, for each chosen desire for which an applicable plan has been obtained, an intention is created and associated with that desire. This intention is then associated with the plan that is expected to fulfill it as well, effectively "adopting" said plan, which the agent executes (Line 6). For each chosen desire for which no way of attempting to fulfill it has been found, mark it as "unsatisfiable at this time" and refrain from creating an intention for it in the current cycle (Line 7). If there is an adopted plan and it fails, then the agent may either seek an alternate plan or give up on the corresponding intention altogether (Line 8). This is subject to a *commitment policy* that may take into account whether this happened before to the desire associated with this intention, to intentions in general (there could conceivably be some kind of overall environment issue behind the failures), the rate at which alternate plans have proven effective, the computational cost for obtaining such plans – perhaps compared to the cost of desire selection, etc.. Successful plan executions cause their associated intentions to be removed (Line 9).

4.5 Beliefs

We consider the belief base to correspond to the entire Bayesian Network, as do Kieling and Vicari (KIELING; VICARI, 2010). We say that there is a belief that there is a probability p that a certain event variable x is in state s. Each event variable has n possible states, each with an associated probability, that either is readily available from a *Conditional Probability Table* if the state of all the conditioning variables is known (i.e., there is evidence on each of them) or has to be obtained (i.e., calculated) via Bayesian inference.

4.6 Desires

Desires in the Bayesian BDI agent model refer to specific states of event variables in the Bayesian Network. Each desire has a preconditioning belief, following the tradition of many implemented BDI systems (e.g., (RAO, 1996), (MÓRA et al., 1999)).

While considering the concept of desires in tandem with Bayesian Networks, we envisioned two diverging approaches. Since in the original BDI model desires may be considered what the agent would like to be true, which would in turn, upon fulfillment, be reflected in its beliefs, we considered extend this notion for Bayesian BDI agents by considering "that which is true" to be interchangeable with "that which is itself backed by hard evidence". This would essentially mean that although the beliefs might be uncertain, the status of each desire would not.

However, as this is a considerable constraint, we believe that it should not be embraced lightly. The concept of a desire (a desired state for a variable) that could be considered satisfied without *true confirmation* (i.e., hard evidence), possibly by means of a threshold-based probability evaluation (e.g., an 80% probability that you do not have a certain disease, based on test results, does not preclude a false negative, but does not pose a figure easy to disregard either), might allow for more flexible agents, *but* such agents would sometimes be "out of touch with reality", believing in and considering themselves to have achieved supposed (desired) facts that would simply not be true.

In an actual human being, this trait might make the individual be branded as delusional. On the other hand, this noisy relationship with the environment does potentially allow for greater efficiency, as it encompasses assumptions rather than depend on confirmations – that might range from being absolutely infeasible to being wildly costly. As such, a counter-argument to the previous delusion label might be that a certain level of this permissive attitude is essential so that there is not a condition that might be described as a "compulsive-obsessive disorder", as requiring confirmation for every desire may be incompatible with realistic scenarios. Given the two points of view, we believe that the calibration of the level of tolerance for inner uncertainty in the desire satisfaction evaluation process is the factor essentially regulating sane (i.e., functional) behavior.

We propose two desire categories differing in satisfaction evaluation: *strong* desires and *weak* desires.

4.6.1 Strong desires

Strong desires are desires on which there must be *hard evidence* so that they can be considered fulfilled. There may not be any doubt, however small, whether or not a strong desire has been satisfied. This is a special case of what would otherwise be a weak desire (described in Section 4.6.2), where, given a desire (i.e., a state of an event variable) d, $P(d) = 1$.

So, while in this agent model there is a concern with supporting uncertainty, the latter is not acceptable in the agent's desire satisfaction evaluation process for this type of desire.

4.6.2 Weak desires

Weak desires are those that are not necessarily expected to be confirmed, via hard evidence, but are expected to be believed to be sufficiently likely to be true. Weak desires may be viewed as desires that accept *soft evidence* as sufficient in order to be considered satisfied. The satisfaction evaluation process for weak desires is threshold-based.

4.7 Intentions

Similarly to traditional BDI agents, intentions are desires to the fulfillment of which the agent has committed itself. The agent will seek a plan – a sequence of actions – that is applicable to the current situation: any plan that is aimed at satisfying at least one of the intentions and whose preconditions are not conclusively denied (i.e., preconditions that are not contradictory to hard evidence) is valid.

We believe that it is paramount to be able to determine whether an action plan has succeeded or failed. The state of plan preconditions *may*, however, be uncertain, conceptually increasing the susceptability to failure during execution. In other words, since uncertainty is a constant factor to be aware of in the current model, plan failure resulting from it should be seen as *imminent*.

Ideally, a plan-generating algorithm that worked with this agent model would be available, along with a plan library. This is a subject outside the scope of this work, however, and here the most practical solution is to presume that all plans are pre-defined.

It is worth noting that whether the preconditions have been confirmed (i.e., supported by hard evidence) or not may be relevant in the event of a plan execution failure, as such a failure may result from the fact that the plan should have never actually been used in the first place (though we were not aware of this at the time of its adoption) or not.

4.8 Example

The *Burglary or Earthquake* Bayesian Network (Figure 4.1) helps illustrate an example. The network in question corresponds to the beliefs of agent BoE. If Watson (e.g., another agent) calls BoE, it is very likely that the alarm has fired off – regardless of cause – as can be seen in the *Conditional Probability Table* for variable $WatsonCalls$.

We define a strong desire $D1(Alarm = false)$ that is conditioned on $Alarm = true$, i.e., a desire to have the $Alarm$ turned off (in other words, to believe the alarm to be off). Our agent believes it highly probable (94.2%) that the alarm is already off under normal circumstances – Bayesian inference with an empty evidence set reflects this:

$$P(Alarm) = \sum_{Earthquake,Burglary} P(Alarm, Earthquake, Burglary)$$

$$=\textstyle\sum_{Earthquake,Burglary} P(Alarm|Earthquake,Burglary)P(Earthquake,Burglary)$$

$$=P(Alarm|Earthquake{=}true,Burglary{=}true)P(Earthquake{=}true,Burglary{=}true)+$$

$$P(Alarm|Earthquake{=}true,Burlgary{=}false)P(Earthquake{=}true,Burglary{=}false)+$$

$$P(Alarm|Earthquake{=}false,Burglary{=}true)P(Earthquake{=}false,Burglary{=}true)+$$

$$P(Alarm|Earthquake{=}false,Burglary{=}false)P(Earthquake{=}false,Burglary{=}false)$$

$$=(0.99,0.01)\cdot(0.01\cdot0.05)+(0.1,0.9)\cdot(0.01\cdot0.95)+(0.95,0.05)\cdot(0.99\cdot0.05)+(0.01,0.99)\cdot(0.99\cdot0.95)$$

$$=(0.99,0.01)\cdot0.0005+(0.1,0.9)\cdot0.0095+(0.95,0.05)\cdot0.0495+(0.01,0.99)\cdot0.9405$$

$$=(0.000495,0.000005)+(0.00095,0.00855)+(0.047025,0.002475)+(0.009405,0.931095)$$

$$=(0.057875,\mathbf{0.942125})$$

Do note that $P(Earthquake, Burglary) = P(Earthquake) \cdot P(Burglary)$ because $Earthquake$ and $Burglary$ are *d-separated* in their *converging* connection towards $Alarm$, i.e., they are conditionally independent given no hard evidence on any of their mutual descendants.

We also define a strong desire $D2(Burglary = false)$, conditioned on $Alarm = true$. This means, as a simple matter of agent design decision, that our agent only cares about determining that there has not been a $Burglary$ as much as it believes that the $Alarm$ has been activated.

Agent BoE possesses an action plan $P1$, that involves going home ("home" being a location that can be computationally interacted with) and determining for sure the state of the alarm, as well as whether someone broke into the house, i.e., obtaining evidence on the current state of variables $Alarm$ and $Burglary$. This illustrates how an action plan is meant to bring forth evidence, so a desire may be satisfied or not. Plan $P1$ may serve both desires, should they be selected, letting corresponding intentions be thereby created.

The actual selection of $D1$ and $D2$ must not depend on hard evidence that $Alarm = true$, as this may simply not come to be available. In other words, traditional BDI agent desire selection, relying on simple logical evaluations, does not help us here.

In Chapter 5 we cover Bayesian BDI Desire Selection.

5 BAYESIAN BDI DESIRE SELECTION

5.1 Introduction

As mentioned in Chapter 2, traditional BDI agents select desires for adoption as intentions based on the state of their belief base, and the constraints that indicate whether a given desire is viable and passive to be adopted as an intention may be expressed as logical expressions over beliefs, with these being commonly expressed as ground literals. This approach, however, does not have the ability to deal with uncertain information.

In Bayesian BDI agents, beliefs are expressed as states of event variables in Bayesian Networks rather than as ground literals, leading to the need for more elaborate mechanisms for desire selection. Fagundes et al.(FAGUNDES; VICARI; COELHO, 2007) propose one such mechanism. In this dissertation, we present a limitation that mechanism possesses, and provide BDI desire selection mechanisms that fit Bayesian BDI agents and do not present said limitation, i.e., Objective 5 (the objectives are listed in Section 1.1).

5.2 Beliefs and Desires

In traditional BDI agents, a desire is commonly represented as a logical rule (e.g., $belief\text{-}formula \Rightarrow desire$). We represent it as follows:

```
<desire> : { <belief> } ,
```

where *desire* denotes the name of the desire and *belief* is a belief that corresponds to a condition *necessary* for this desire to be considered conceptually adequate for selection. Each belief is generally a ground literal, which is then evaluated to either *true* or *false*.

Bayesian BDI agents' beliefs are extended with probabilistic data – each belief corresponds to a state of an event variable in a Bayesian Network with which a probability is associated. Thus, it is no longer sufficient to perform the previous logical evaluation of required beliefs for each desire potentially eligible for intention creation; in other words, for Bayesian BDI agents, a procedure devised for traditional BDI agents for determining whether the condition that indicates whether or not a desire should be selected to have an associated intention has been met is limited in a probabilistic context, as it does not differentiate between the degrees of probability associated with desire preconditions. The only case where desire preconditions still constitute a matter of validity is when there is hard evidence *against* the precondition in question (i.e., evidence of a different state of the event variable). Also, just as there are degrees of probability in the beliefs, selecting a desire is now a decision made with varying degrees of confidence, which implies that preconditions are no longer strictly about *validity* of selection, but also about *confidence* in a selection that is inherently rooted in uncertainty.

5.3 Desire Selection

Let us consider a hypothetical Bayesian BDI agent $A1$, possessing a Bayesian Network $BN1$ that represents its beliefs and a set of desires $\{D_1, D_2, D_3\}$, all associated with beliefs (e.g., $desires = \{\ D_1(B_1), D_2(B_2), D_3(B_3)\ \}$) expected to be true in order for the desire to be valid. Unless there is evidence on a state of the event variable referred to by the precondition, we do not know if selecting a desire is truly appropriate. Nevertheless, we must proceed in the midst of this uncertainty.

We present algorithms for desire selection in both pseudocode and in natural language. The pseudocode resembles Algorithm 3, a simple selection algorithm. Concerning the arguments listed (Line 1), $evaluationCriterion$ is an (unspecified) evaluation criterion, and $availableDesires$ is the list of available desires. Each element of $availableDesires$ is represented as a variable $desire$. This particularly vague sample algorithm traverses the list of available desires (Line 2); for each one of them, the algorithm evaluates it (Line 3). If the desire currently being considered is found eligible for selection (Line 4), it is removed from the list of available desires (Line 5) and returned (Line 6), implying that the algorithm may choose at most only one desire. If the entire list of available desires is traversed and no desire has been found eligible for selection (Line 8), the algorithm returns $null$ (Line 9), indicating that there are no desires selected (i.e., the desire selection *failed*).

If the desire selection succeeds, this means that it caused the selected desire to be removed from the list of available desires, so it may not be chosen while there is an intention being tentatively created (pending on plan availability) or while there is an intention associated with it. It is very important to point out that *once an intention has been dropped* (i.e., not fulfilled)*, the desire is added back to the list of available desires*; without this, failed desires would be lost. This is not shown in any of the selection algorithms, as it is not a part of desire selection itself, but is a key consideration for whenever the aforementioned desire removal occurs in them.

Algorithm 3 Simple Selection

```
1:  procedure SimpleSelection($evaluationCriterion$, $availableDesires$)
2:      for each $desire$ such that $desire \in availableDesires$ do
3:          perform evaluation of $desire$ following $evaluationCriterion$
4:          if $desire$ is eligible for selection then
5:              $availableDesires.remove(desire)$
6:              return $desire$
7:          end if
8:      end for
9:      return $null$
10: end procedure
```

Throughout this chapter, we detail selection algorithms also by presenting examples. We adopt both a simple and an extended agent code language for describing agents to be used in examples. The reason for this is that the simple version refrains from specifying the Bayesian Network representing the agent's beliefs, so as to emphasize the desires and their preconditions, while the extended version covers both the Bayesian Network – the beliefs – and the desires. Do note that *<IDENTIFIER>* refers to a literal beginning with a letter and that may contain letters and numbers, and *<PROBABILITY>* refers to a numerical value in the $[0, 1]$ interval. Comments are preceded by two consecutive forward slashes (i.e., *//*). The following is the simple abstract agent code language definition:

```
agent -> "agent" <IDENTIFIER> "{" agent_body "}"
agent_body -> beliefs desires
beliefs -> "beliefs" "=" <IDENTIFIER> ";"
desires -> "desires" "{" desire_body "}"
desire_body -> desire_definition*
desire_definition -> <IDENTIFIER> "{" <IDENTIFIER> "}"
```

The simple agent code language definition begins with the keyword “agent”, followed by a name meant to identify it (i.e., an *identifier*) and an agent body, which must be enclosed in curly brackets (i.e., *{ }*). An agent body comprises the beliefs and desires. The beliefs are informed using the keyword “beliefs”, an equal sign (i.e., =), then a Bayesian Network identifier, and finally a semicolon (i.e., *;*); this conveys that the beliefs are assigned a Bayesian Network, the latter not being specified in this simple pseudocode language. Desires are informed using the keyword “desires”, followed by a desire body, which is enclosed in curly brackets. A desire body comprises n desire definitions. Each desire definition comprises a desire identifier, followed by a belief identifier – referring to the desire’s precondition, enclosed in curly brackets.

Next is agent code describing agent $A1$:

```
agent A1 {

    beliefs = BN1;

    desires {
        D1 { B1 }
        D2 { B2 }
        D3 { B3 }
    }
}
```

The code tells us that we have an agent identified as $A1$, whose beliefs correspond to Bayesian Network $BN1$ and whose desires consist of D_1, conditioned on belief B_1, D_2, conditioned on belief B_2, and D_3, conditioned on belief B_3.

Now we will present the definition for the extended agent code language to be used:

```
agent -> "agent" <IDENTIFIER> "{" agent_body "}"
agent_body -> beliefs desires
beliefs -> "beliefs" bayesian_network
bayesian_network -> <IDENTIFIER> "{" bayesian_network_body "}"
bayesian_network_body -> event_variable_definition ("," event_variable_definition)*
event_variable_definition -> <IDENTIFIER> "(" event_variable_body ")"
event_variable_body -> list_of_state_definitions
                    | single_parent_event_variable_clause ("," single_parent_event_variable_clause)*
                    | multi_parent_event_variable_clause ("," multi_parent_event_variable_clause)*
list_of_state_definitions -> state_definition ("," state_definition)*
state_definition -> <IDENTIFIER> "=" <PROBABILITY>
single_parent_event_variable_clause -> event_variable_state "(" list_of_state_definitions ")"
multi_parent_event_variable_clause -> "{" event_variable_state ("," event_variable_state)* "}" "(" list_of_state_definitions ")"
event_variable_state -> <IDENTIFIER>"."<IDENTIFIER>
desires -> desire*
desire -> ("strong" | "weak") "desire" event_variable_state "(" event_variable_state ")" ";"
```

The extended agent code language definition begins with the keyword “agent”, followed by an identifier and an agent body. An agent body comprises the beliefs and desires. The beliefs must be preceded by the keyword “beliefs”, and a Bayesian Network definition, which must begin with an identifier and then be specified as a series of event variable definitions. Each event variable definition consists of an identifier and a body. An event variable body must be specified in the one of the following ways:

- If there are no parents, there must be a list of state definitions;

- If there is exactly one parent, list the parent's states, each one accompanied by a list of state definitions for the event variable currently under consideration (i.e., the child) given the parent state; we are talking about a *Conditional Probability Table* (CPT);

- If there are multiple parents, list all the combinations of their states – keeping in mind that only one state is possible for each event variable at a time, each combination accompanied by a list of state definitions for the event variable currently under consideration given that combination (e.g., event variable D in agent $MyBayesianAgent$); this is also a CPT.

Note that the third specification for the event variable body is actually the general form: if there are no parents (the case in the first definition), the set corresponding to all combinations of parent states is empty; if there is a single parent (the case in the second definition), the set is equal to the parent variable's state space (e.g., event variables C and E in agent $MyBayesianAgent$). beh A state definition consists of a state identifier, an equals sign and a probability value; in other words, it expresses that the referred state is defined as being associated with the informed probability given the specified parent states, which may be none. Next, desires are expected. First, either the "strong" or the "weak" keyword must be present [1], indicating whether the desire that follows is strong or weak, then the keyword "desire", followed by an event variable state representing a desire, and an event variable state, in parentheses, representing its precondition. An event variable state is composed of an event variable identifier (its "name"), a dot and a state identifier.

Given the extended agent code language, another agent we will use to illustrate possible execution cycles for desire selection is described as follows:

```
agent MyBayesianAgent {
    beliefs BN {
        A(true = 0.01, false = 0.99), // unconditioned probabilities
        B(true = 0.05, false = 0.95), // unconditioned probabilities
        C(A.true(true = 0.9, false = 0.1),
          A.false(true = 0.01, false = 0.99)), // CPT
        D({B.true, A.true}(true = 0.99, false = 0.01),
          {B.true, A.false}(true = 0.95, false = 0.05),
          {B.false, A.true}(true = 0.1, false = 0.9),
          {B.false, A.false}(true = 0.01, false = 0.99)), // CPT
        E(D.true(true = 0.9, false = 0.01),
          D.false(true = 0.01, false = 0.99)) // CPT
    }

    strong desire D.false(E.true);

    strong desire B.false(D.true);
}
```

The code tells us that we have an agent $MyBayesianAgent$, whose beliefs correspond to Bayesian Network BN and whose desires consist of strong desires $D.false$ (i.e., the wish to attain hard evidence that $D = false$) and $B.false$, which are conditioned on beliefs $E.true$ (i.e., the belief associated with state $true$ of event variable E in the Bayesian Network) and $D.true$, respectively. BN is defined as a Bayesian

[1]As seen in Section 4.6, desires can be either *strong* or *weak*, affecting the level of confirmation (the associated probability) that the agent will find acceptable in order to consider them fulfilled.

Network with five event variables, A, B, C, D and E, each of which has two states: $true$ and $false$ (i.e., boolean states). The state definitions for all the event variables are provided, given each parent state combination. The parent state combination sets presented alongside the lists of state definitions tell us that A and B have no parents, C is the child of A, D is the child of A and B, and E is the child of D. Examples of probabilities expressed in BN include $P(A = true) = 0.01$, $P(A = false) = 0.99$, $P(C = false|A = true) = 0.1$, $P(D = true|A = true, B = false) = 0.1$, $P(D = false|A = false, B = true) = 0.05$ and $P(E = false|D = false) = 0.99$.

In addition to the agent definitions, we will represent the agent's beliefs at a given moment by listing the event variables along with their states and probabilities, as in the example below:

```
initial beliefs = { // no hard evidence available
    A(true = 0.01, false = 0.99),
    B(true = 0.05, false = 0.95),
    C(true = 0.0189, false = 0.9811), // computed
    D(true = 0.057875, false = 0.942125), // computed
    E(true = 0.06150875, false = 0.93849125) // computed
}
```

All probability values are computed and presented. $P(A) = (0.01, 0.99)$, $P(B) = (0.05, 0.95)$, $P(C) = (0.0189, 0.9811)$, $P(D) = (0.057875, 0.942125)$ and $P(E) = (0.06150875, 0.93849125)$. In this case, we presented the *initial* beliefs; there is no evidence on any state of any event variable at the moment.

5.3.1 Example Considerations

Specifically for the sake of illustrating algorithmic execution, we will:

- assume that, for agent $A1$, the following probabilities have been calculated in $BN1$: $P(B_1) = 0.4$, $P(B_2) = 0.75$, $P(B_3) = 0.1$;
- assume two plans for $MyBayesianAgent$: $P1$ and $P2$. $P1$ possibly succeeds in changing the environment and/or bringing about evidence on event variable D, whereas $P2$ does that for event variable B.

It is worth noting that a plan may fail during execution, and that the selection cycles presented in the following sections are examples of *possible* selection cycles; whether an adopted plan fails or not, and, where applicable, numeric values that are randomly generated are *nondeterministic* elements in scenarios.

Desires preconditioned on beliefs holding a probability that is exactly equal to zero, i.e., as a result of hard evidence on a state other than the one referred to by the belief in question, but associated with the same event variable, are skipped.

5.3.2 Threshold-based Selection

To allow an agent to manage desire selection with uncertainty, Fagundes et al. (FAGUNDES; VICARI; COELHO, 2007) propose selecting desires following a threshold-based criterion where a desire is eligible for selection if the evaluated probability (e.g., the one associated with a preconditioning belief) is greater than, or equal to a given numeric threshold value. This is a conservative approach, designed to disregard desires supported by low probabilities, a principle which is nevertheless a (self-imposed) limitation, as there might be desires that consistently fail to satisfy such a criterion, yet have the potential to actually bring about scenarios interesting to the agent.

Algorithm 4 contains pseudocode representing a threshold-based selection algorithm in the context of a Bayesian BDI agent. It expects the numeric threshold value and the list of available desires (Line 1). The algorithm traverses the list of available desires (Line 2); for each desire therein, the algorithm evaluates whether the probability of its associated precondition is greater than, or equal to the threshold (Line 3). If so, the desire in question is removed from the list of available desires (Line 4) and returned (Line 5), thereby refraining from continuing to traverse the list, the implication being that this algorithm selects at most one desire. If the entire list of available desires is traversed and no desire has been selected (Line 7), the algorithm returns $null$ (Line 8).

Algorithm 4 Threshold-Based Selection

```
1: procedure ThresholdBasedSelection($threshold$, $availableDesires$)
2:     for each $desire$ such that $desire \in availableDesires$ do
3:         if $desire.preCondition.probability \geq threshold$ then
4:             $availableDesires.remove(desire)$
5:             return $desire$
6:         end if
7:     end for
8:     return $null$
9: end procedure
```

Considering agent $A1$, let us analyze two possible scenarios:

Example 5.1: agent $A1$ and threshold-based selection.

```
P(B₁) = 0.4
P(B₂) = 0.75
P(B₃) = 0.1

D₁ { B₁ }
D₂ { B₂ }
D₃ { B₃ }
```

- Scenario #1: $threshold = 0.7$

Only D_2 may be chosen, as the probability of its precondition – B_2 – is the only one greater than, or equal to the threshold ($0.75 \geq 0.7$) among the desire preconditions. This might be sensible, as a 70+% requirement for a belief to be worth considering is not particularly strict, nor is it excessively permissive. However, a belief holding a lower probability obviously only denotes a smaller chance of its associated event variable state being true, not the impossibility of this fact. It might also be case that there is no known way for the agent to possibly obtain a probability higher than 40% for B_1 in order to make the selection of B_1 a more promising decision. Does this mean that D_1 must absolutely not be selected? We will lower the threshold to accomodate D_1 next.

- Scenario #2: $threshold = 0.4$

Given this significantly permissive probability threshold validating beliefs, D_1 may now be chosen (40% probability held by B_1), as well as D_2 (75% probability held by B_2). Agent $A1$ will clearly intend to satisfy not only D_2, but also D_1.

Next is an example using $MyBayesianAgent$:

Example 5.2: $MyBayesianAgent$ *and threshold-based desire selection.*

```
initial beliefs = { // no hard evidence available
    A(true = 0.01, false = 0.99),
    B(true = 0.05, false = 0.95),
    C(true = 0.0189, false = 0.9811), // computed
    D(true = 0.057875, false = 0.942125), // computed
    E(true = 0.06150875, false = 0.93849125) // computed
}


threshold = 0.8

Desires:
    D.false { E.true (0.06150875) }
    B.false { D.true (0.057875) }
```

As no desire is backed by a belief holding a probability equal to, or greater than the defined threshold, no desire is selected.

One might argue that a dynamic threshold meant to alleviate or increase strictness, bound by some criteria, is the answer. In any case, there does seem to be room for improvement. What if there is no prior experience to use as statistical reference, or if the environment changes considerably at a fast rate that thwarts prediction?

In order to select desires conditioned on uncertain beliefs, we believe that selection based on a given probability threshold is insufficient, because there may be a subset of desires that will never satisfy such a requirement. In what follows, we present alternative desire selection approaches aimed at addressing this limitation.

5.3.3 Probability Ranking

Our first approach involves sorting the desire list by precondition probability, in decreasing order; this way we have a ranking ranging from the desire with the precondition holding the highest probability to the desire with the precondition holding the lowest probability. We then pick the first desire in the list (i.e., the desire backed by the belief most likely to be true), and create an intention for it. This intention is associated with a plan, obtained either from a plan library or on-the-fly using a planning algorithm.

Algorithm 5 contains pseudocode representing the Probability Ranking selection algorithm. It expects a list of the available desires ranked according to the probability of their preconditions, expressed as $rankedDesires$ (Line 1). A list's $length$ property refers to its element count. If the $length$ of $rankedDesires$ is greater than zero, i.e., if $rankedDesires$ is not empty (Line 2), the algorithm selects the first desire (Line 3) and, if the probability of that desire's precondition is greater than 0 (Line 4) – to prevent a desire associated with a contradicted precondition from being selected (in conformance with what is said in Section 5.3.1), removes that desire from the list (Line 5) and returns it (Line 6). Otherwise, the algorithm returns $null$ (Line 9). Do note that $rankedDesires$ is expected to be maintained outside the selection procedure itself. One way of accomplishing this is having a procedure responsible for updating it be called whenever there is a belief update (i.e., a change in the Bayesian Network, in the case of Bayesian BDI agents), so as to keep $rankedDesires$ ready to be used in the selection procedure.

Algorithm 5 Probability Ranking Selection

```
1: procedure ProbabilityRankingSelection($rankedDesires$)
2:     if $rankedDesires.length > 0$ then
3:         $desire \leftarrow rankedDesires.first()$
4:         if $desire.preCondition.probability > 0$ then
5:             $rankedDesires.remove(desire)$
6:             return $desire$
7:         end if
8:     end if
9:     return $null$
10: end procedure
```

The function in Algorithm 6 illustrates the comparison to be performed in whatever sorting algorithm one may choose to employ for $rankedDesires$. The sorting order is directly tied to the probability associated with each desire's precondition. Given two desires, $d1$ and $d2$ (Line 1), Function $CompareDesires$ returns 1 (Line 3) if $d1$ has a higher sorting order than $d2$ (Line 2), -1 (Line 5) if $d1$ has a lower sorting order than $d2$ (Line 4), or 0 (Line 7) if they have the same sorting order (Line 6).

Algorithm 6 Probabilistic Desire Pre-Condition Comparison

```
1: function CompareDesires($d1$, $d2$)
2:     if $d1.preCondition.probability > d2.preCondition.probability$ then
3:         return $1$
4:     else if $d1.preCondition.probability < d2.preCondition.probability$ then
5:         return $-1$
6:     else
7:         return $0$
8:     end if
9: end function
```

Thanks to the uncertainty in the environment, the possibility of a Bayesian BDI agent's intentions failing, including, but not exclusively due to incorrect belief-related assumptions, must be taken into account. What happens in this case is subject to the policy adopted by the agent – the following are examples of how a Bayesian BDI agent might behave:

- An agent can be "stubborn" and keep the desire selected, hoping that the intention does not fail the next time. In this case, it is wise to impose a limit to how many times a desire associated with an intention that just failed may be kept in detriment to the next desire in the list, so that we do not deliberately run into the "desire never getting chosen" problem. Note that insisting on a desire that fails to be achieved due to false assumptions on its preconditions (i.e., a desire selection that would not be made were it not for the uncertainty of the preconditioning beliefs) is expected to keep resulting in failures;

- The desire that failed to be fulfilled can be moved down the list a certain number of positions according to a pre-defined formula, allowing the next desire in the list to be picked;

- Desires can each have an intention failure counter or be stored in lists each associated with a certain number of failures and such can serve as a criterion for deciding the next desire to be picked;
- Desires that failed to be fulfilled can simply be inserted at the end of the list, thereby giving a chance to the previously lowest-ranked desire; this is the most straightforward approach among the ones listed, and we take it as the default policy in the scope of this chapter unless otherwise specified.

Do note that despite the fact that here the desire list is originally structured as a probability ranking, these strategies make it so that the ordering does not have to respect the desire precondition probability criterion at all times. For instance, a desire preconditioned on the belief holding the highest probability might be at the very end of the list at a given time as the result of a plan failure that led to its associated intention being dropped.

We will now proceed to presenting examples.

Example 5.3: agent $A1$ and Probability Ranking. For agent $A1$, we obtain a desire ranking as follows:

```
format:
    <desire> { <preconditioning belief> (<current belief probability>)) }

desire ranking:
    D₂ { B₂ (0.75) }
    D₁ { B₁ (0.4) }
    D₃ { B₃ (0.1) }
```

where D_2 is the highest-ranked desire, supported by a belief (B_2) holding a 75% probability, and D_3 is the lowest-ranked desire, supported by a belief (B_3) holding a 10% probability. The desire selection process must first choose D_2 for there to be created an associated intention, since it is the desire most likely to be viable in the uncertain environment agent $A1$ is situated in. Do realize that D_3 *will* be chosen once D_2 and D_1, in this order, have been chosen and treated appropriately, that is, by potentially having corresponding intentions generated (pending on the availability of applicable plans) and fulfilled.

If an adopted plan fails, so may the intention, or a new plan may be sought. If it succeeds, so do both the intention and the fulfillment of its associated desire. Especially if there is no current intention, we must run the desire selection procedure again soon. Thanks to this ranking, even the desire with the smallest probability of being viable will be selected at some point, leading to an intention aimed at fulfilling it (i.e., no desire is absolutely irrelevant).

As no desire is backed by a belief holding a probability equal to, or greater than the defined threshold, no desire is selected.

Next is an example using $MyBayesianAgent$:

Example 5.4: $MyBayesianAgent$ and Probability Ranking.

```
initial beliefs = { // no hard evidence available
    A(true = 0.01, false = 0.99),
    B(true = 0.05, false = 0.95),
    C(true = 0.0189, false = 0.9811), // computed
    D(true = 0.057875, false = 0.942125), // computed
```

```
        E(true = 0.06150875, false = 0.93849125) // computed
    }

    Desire list (ranking):
        1. D.false (P(E.true) = 0.06150875)
        2. B.false (P(D.true) = 0.057875)
```

The probability ranking in this case consists of two desires backed by beliefs holding very low probabilities: $D.false$ is conditioned on $E.true$, whose probability is 0.06150875, and $B.false$ is conditioned on $D.true$, whose probability is 0.057875.

Selection cycle #1: desire $D.false$ is selected. Plan $P1$ is selected, as it is the only plan that may bring about hard evidence that event variable D is in state $false$. It should be noted that this happens despite the very low probability held by the precondition $E.false$.

$P1$ fails, therefore no evidence is obtained. Desire $D.false$ is placed at the end of the list.

```
    Desire list:
        1. B.false (P(D.true) = 0.057875)
        2. D.false (P(E.true) = 0.06150875)
```

Selection cycle #2: desire $B.false$ is selected. Plan $P2$ is selected, as it is the only plan that may bring about hard evidence that B is in state $false$. This happens with an even lower precondition probability than in the previous cycle. At least we can see that even particularly unpromising desires get a chance when desire selection is performed in this manner. The ease with which this occurs may be questionable, however.

$P2$ is executed and the agent obtains evidence of $B.true$. This means that the intention could not be satisfied at this time using $P2$. There are no other plans that might prove useful, and the intention is dropped.

```
    current beliefs = { // hard evidence that B = true
        A(true = 0.01, false = 0.99),
        B(true = 1.0, false = 0.0),
        C(true = 0.0189, false = 0.9811), // computed
        D(true = 0.9504, false = 0.0496), // computed
        E(true = 0.85585, false = 0.14415) // computed
    }
```

The probabilities associated with the desire preconditions have changed. However, it is this agent's policy that desire $B.false$ nonetheless be moved to the end of the list due to the recent intention failure, taking priority over what would constitute a true probability ranking at this point (one where $B.false$ would keep its position before $D.false$, as the probability of their preconditions would suggest).

```
    Desire list:
        1. D.false (P(E.true) = 0.85585)
        2. B.false (P(D.true) = 0.9504)
```

Selection cycle #3: desire $D.false$ is selected again. $P1$ is adopted for the new intention. $P1$ succeeds and evidence of $D.false$ is obtained, satisfying the intention and subsequently the desire itself.

```
current beliefs = { // hard evidence that B = true and D = false
    A(true = 0.00202, false = 0.99798), // computed
    B(true = 1.0, false = 0.0),
    C(true = 0.01179, false = 0.98821), // computed
    D(true = 0.0, false = 1.0),
    E(true = 0.01, false = 0.99) // computed
}

Desire list:
    1. B.false (P(D.true) = 0.0)
```

Selection cycle #4: this is a specific situation where the precondition for desire $B.false$ is *known* not to be satisfied (see Section 5.3.1). Given this, the desire is not selected.

As there are no other desires, the agent will keep considering and skipping desire $B.false$ in future selection cycles, until the probability of its precondition is evaluated to a value greater than zero.

Now, one issue with the approach used in Probability Ranking selection is that the cost of pursuing an intention (i.e., executing a plan associated with an intention) may be considerable, and desires quite unlikely to be satisfied may be easily chosen, although obeying a priority ordering. This is something we realized while considering possible agent scenarios. In order to offer a solution that seems more reasonable and is still not hindered like the previously proposed threshold solution, we explored yet another possibility.

5.3.4 Biased Lottery

This approach entails pushing the very notion of probability – a notion encompassing this work – into the desire selection process itself. The idea is to randomly generate a number and use it to determine which desire to choose, according to a probability distribution appropriately reflecting the probabilities of the desires' preconditions. We generate accordingly-sized numeric intervals, all in $[0, 1]$. If the sum of the probabilities of the beliefs supporting the desires is greater than 1, we normalize them. The probabilities serve as weights that create bias in what would otherwise constitute a purely random selection; it is a nondeterministic desire selection that is subject to bias from the probability distribution. This desire selection method neither disregards desires backed by beliefs holding very low probabilities, nor is designed to embrace them more often than common sense would permit – than such probabilities would suggest.

Algorithms 7 and 8 contain pseudocode representing the Biased Lottery desire selection, the latter presenting a function called by the former.

Algorithm 7 expects the list of available desires (Line 1). It generates a random numeric value (Line 2), and then generates a list of numeric values (Line 3) that correspond to the upper limits (boundaries) for the numeric intervals to be used for determining which desire must be selected, if any; variables $randomValue$ and $intervals$ refer to the random numeric value and the list of upper limits for the intervals, respectively; Function $GenerateIntervals$ (Line 3) is detailed in Algorithm 8.

Algorithm 7 proceeds to traverse the list of upper interval limits (Line 4); for each such numeric value, if the random value is smaller than it (Line 5), the algorithm selects the desire associated with the numeric interval in question (Line 6), removes that desire from the list of available desires (Line 7), and returns the desire (Line 8). Note that the list of upper interval limits obeys an ascending order, and that is the reason that the described evaluation (Line 5) works. If the entire list of upper interval limits is traversed and the random value has not been found to belong to any of the intervals (Line 10), the algorithm returns $null$ (Line 11).

Algorithm 7 Biased Lottery

```
1:  procedure BIASEDLOTTERY($availableDesires$)
2:      $randomValue \leftarrow GenerateRandomNumber()$
3:      $intervals \leftarrow GenerateIntervals(availableDesires)$
4:      for $i \leftarrow 0$ to $intervals.length$ do
5:          if $randomValue < intervals[i]$ then
6:              $desire \leftarrow availableDesires[i]$
7:              $availableDesires.remove(desire)$
8:              return $desire$
9:          end if
10:     end for
11:     return $null$
12: end procedure
```

Algorithm 8 expects a list of desires (Line 1). The algorithm starts by creating $intervals$ – a list meant to store the upper numeric interval limits that will be calculated (Line 2). Provided that there are elements in the supplied desire list, i.e., that the $length$ of $desires$ is greater than 0 (Line 3), the algorithm proceeds to create a list – $probabilities$ – that will store the probabilities of the desires' preconditions (Line 4). It also defines a variable sum that will be used to store the sum of all such probabilities, initializing it to 0 (Line 5). It then traverses the desire list (Line 6); for each desire, the probability of its precondition is assigned to the corresponding position in $probabilities$ (Line 7), and sum is incremented by this probability (Line 8). After the aforementioned traversal, the algorithm evaluates whether the sum of probabilities is greater than 1 (Line 10), which is the criterion for deciding to perform normalization. If it is, then (Line 10) the first position of $intervals$ is assigned the normalized probability of the first desire's precondition, i.e., the probability stored in $probabilities[0]$ – the first position of $probabilities$ – divided by the sum of all the probabilities stored therein (Line 11). In addition to this, for each succeeding position of $intervals$ – whose $length$ is the same as the number of desires, the algorithm assigns it the value of its preceeding position plus the probability stored in the same position in $probabilities$, but *normalized* (Line 13). If the normalization evaluation (Line 10) was false, then (Line 15) the first position of $intervals$ is assigned the probability of the first desire's precondition, i.e., $probabilities[0]$ (Line 16). Also, for each succeeding position of $intervals$, the algorithm assigns it the value of its preceeding position plus the probability stored in the same position in $probabilities$ (Line 18). Note that Lines 11 and 13 are the normalized equivalents of Lines 16 and 18, and all of these calculate and ultimately assign upper interval limits to the positions in $intervals$, gradually populating the latter. Once all the positions in the list of upper interval limits have been assigned values, the list is returned (Line 22). If there are no desires (Line 3), the list remains empty and is returned as such (Line 22).

For the next example, let us consider agent $A1$.

Example 5.5: agent $A1$ and Biased Lottery. The sum of the probabilities held by the desires' preconditions is $0.4 + 0.75 + 0.1 = 1.25 > 1.0$, therefore we have to normalize

Algorithm 8 Biased Lottery – Desire Intervals

```
1: function BIASEDLOTTERY→GENERATEINTERVALS($desires$)
2:     $intervals[desires.length]$
3:     if $desires.length > 0$ then
4:         $probabilities[desires.length]$
5:         $sum \leftarrow 0$
6:         for $i \leftarrow 0$ to $desires.length$ do
7:             $probabilities[i] \leftarrow desires[i].preCondition.probability$
8:             $sum \leftarrow sum + probabilities[i]$
9:         end for
10:        if $sum > 1$ then
11:            $intervals[0] \leftarrow \frac{probabilities[0]}{sum}$
12:            for $i \leftarrow 1$ to $intervals.length$ do
13:                $intervals[i] \leftarrow intervals[i-1] + \frac{probabilities[i]}{sum}$
14:            end for
15:        else
16:            $intervals[0] \leftarrow probabilities[0]$
17:            for $i \leftarrow 1$ to $intervals.length$ do
18:                $intervals[i] \leftarrow intervals[i-1] + probabilities[i]$
19:            end for
20:        end if
21:    end if
22:    return $intervals$
23: end function
```

those probabilities: $\frac{0.4}{1.25} = 0.32$, $\frac{0.75}{1.25} = 0.6$ and $\frac{0.1}{1.25} = 0.08$. We use the normalized probabilities 0.32, 0.6 and 0.08 as the sizes for the numeric intervals to be associated with D_1, D_2 and D_3.

```
Probabilities held by the beliefs:
    P(B1) = 0.4
    P(B2) = 0.75
    P(B3) = 0.1

Desires and preconditioning beliefs:
    D1 { B1 }
    D2 { B2 }
    D3 { B3 }

Desire probability distribution intervals: (the desire order is irrelevant)
    D1: [0, 0.32)
    D2: [0.32, 0.92)
    D3: [0.92, 1]
```

Now each randomly generated real numeric value in the $[0, 1]$ interval will result in one of the desires being chosen, respecting the degrees of certainty for their respective preconditioning beliefs.

Next is an example using $MyBayesianAgent$:

Example 5.6: $MyBayesianAgent$ and Biased Lottery.

```
initial beliefs = { // no hard evidence available
    A(true = 0.01, false = 0.99),
    B(true = 0.05, false = 0.95),
    C(true = 0.0189, false = 0.9811), // computed
    D(true = 0.057875, false = 0.942125), // computed
    E(true = 0.06150875, false = 0.93849125) // computed
}
```

```
Desires and preconditioning beliefs:
    D.false { E.true (0.06150875) }
    B.false { D.true (0.057875) }

P(E.true) + P(D.true) = 0.11938375 ≤ 1.0 // no need for normalization

Desire probability distribution intervals:
    D.false: [0, 0.06150875)
    B.false: [0.06150875, 0.11938375)
```

Given desires $D.false$ and $B.false$, conditioned on $E.true$ and $D.true$, respectively, each desire is associated with a numeric interval that starts immediately after the preceeding one ends – the first one starting at 0 – and possesses the same length as the probability of that desire's precondition, except that it is normalized if the sum of all these probabilities exceeds 1. For $D.false$, the corresponding interval is $[0, 0.06150875)$, and for $B.false$, the corresponding interval is $[0.06150875, 0.11938375)$.

- Selection cycle #1: number 0.7632 is randomly generated. As it does not belong to any of the available intervals, no desire is selected.

- Selection cycle #2: number 0.2315 is randomly generated. As it does not belong to any of the available intervals, no desire is selected.

- Selection cycle #3: number 0.014 is randomly generated. It belongs to the interval designated to desire $D.false$, which is therefore selected. Plan $P1$ is chosen and a corresponding intention is thus adopted. Unfortunately, the agent encounters an unexpected error and the plan fails to execute to its completion. As the intention has been met with a plan failure and there are no other applicable plans, the agent abandons it (i.e., it is this agent's policy that it does not insist on the same plan just in case the latter somehow yields a different result the next time).

- Selection cycle #4: number 0.11939 is randomly generated. As it does not belong to any of the available intervals, no desire is selected.

- Selection cycle #5: number 0.1311 is randomly generated. It belongs to the interval designated to desire $B.false$, which is therefore selected. Plan $P2$ is chosen for the new intention. $P2$ is executed and the agent obtains evidence of $B.false$.

```
current beliefs = { // hard evidence that B = false
    A(true = 0.01, false = 0.99),
    B(true = 0.0, false = 1.0),
    C(true = 0.0189, false = 0.9811), // computed
    D(true = 0.0109, false = 0.9891), // computed
    E(true = 0.0197, false = 0.9803) // computed
}
```

A new set of numeric intervals reflecting the probability distributions for the desire preconditions is generated.

```
Desires:
    D.false { E.true (0.0197) }

P(E.true) = 0.0197 < 1.0 // no need for normalization

Desire probability distribution intervals:
    D.false: [0, 0.0197)
```

- Selection cycle #6: number 0.5973 is randomly generated. As it does not belong to any of the available intervals, no desire is selected.

- Selection cycle #7: number 0.9624 is randomly generated. As it does not belong to any of the available intervals, no desire is selected.

- Selection cycle #8: number 0.03092 is randomly generated. As it does not belong to any of the available intervals, no desire is selected.

- Selection cycle #9: number 0.01576 is randomly generated. It belongs to the interval designated to desire $D.false$. The agent successfully executes $P1$ and finds that $D = false$, completing the plan execution, satisfying the intention and, consequently, the hard desire.

```
current beliefs = { // hard evidence that B = false and D = false
    A(true = 0.009099, false = 0.990901), // computed
    B(true = 0.0, false = 1.0),
    C(true = 0.018098, false = 0.981902), // computed
    D(true = 0.0, false = 1.0),
    E(true = 0.01, false = 0.99) // computed
}

No pending desires.
```

We do not perform normalization when the sum of the precondition probabilities is less than 1.0, as this would inflate selection probabilities for desires preconditioned on insignificant events. For example, if there is a single desire, preconditioned on a belief with 0.0001 probability, it would be treated as though its probability were 1.0. Note that the numeric intervals for the desires are forced not to intersect with one another, since the one randomly generated number (per selection cycle) is expected to fit into, at most, one such interval. This is related to the fact that no more than one desire is expected to be selected.

This consideration gave rise to yet another approach to desire selection.

5.3.5 Multi-Desire Biased Random Selection

While working on Biased Lottery, we considered yet another approach to desire selection. Until now, we worked under the assumption that each selection cycle should yield at most one desire. We now consider the thought that it might be a good idea to *select multiple desires at once*.

This approach to desire selection removes competition among desires, so long as they do not conflict, considering such non-conflicting desires independently of each other. We consider a conflict to exist between two desires if they refer to the same event variable, but to different states – all states in the state space of an event variable are mutually exclusive. If a given desire being evaluated conflicts with a desire that is already designated to be selected at the end of the current selection process, we consider it ineligible for selection. Otherwise, it gets a chance: given a desire D_i preconditioned on a belief holding a probability P_i, we say that D_i is assigned a numeric interval $I_i = [0, P_i]$. For every such desire D_i, if a randomly generated numeric value N_i in interval $[0, 1]$ belongs to interval I_i, the desire is added to the set of desires to be selected at the end of this selection cycle. Optionally, we may order the desire list in ascending order of precondition probability, prior to the execution of the algorithm, to let a desire supported by a lower probability have its chance unhindered by potentially conflicting desires that stand a better chance of being selected. If the difference in precondition probabilities of conflicting desires isn't big, however, this may be considered an unwelcome bias, as it could artificially make a supposedly less likely-to-be-selected desire get selected more often than conflicting desire(s) in practice. Another option would be a pure shuffle.

Algorithm 9 contains pseudocode representing the Multi-Desire Biased Random Selection algorithm. The algorithm expects the list of available desires (Line 1), and defines a list – represented as variable $selectedDesires$ – meant to store any number of desires that may be selected (Line 2). It then traverses the list of available desires (Line 3); for each such desire, if it does not conflict with any of the desires already added to $selectedDesires$ (Line 4), the algorithm generates a random numeric value (Line 5) and evaluates if it is smaller than, or equal to the probability of that desire's precondition (Line 6). If it is, then Line 6) the desire in question has been found eligible for selection, and is thus added to the list of selected desires (Line 7) and removed from the list of available ones (Line 8). After the list of available desires has been traversed (Line 11), the list of selected desires is returned (Line 12).

Algorithm 9 Multi-Desire Biased Random Selection

```
 1: procedure MULTIDESIREBIASEDRANDOMSELECTION($availableDesires$)
 2:     $selectedDesires \leftarrow \{\}$
 3:     for each $desire \in availableDesires$ do
 4:         if $desire$ does not conflict with any element in $selectedDesires$ then
 5:             $randomValue \leftarrow random\ number \in [0, 1]$
 6:             if $randomValue \leq desire.preCondition.probability$ then
 7:                 $selectedDesires.add(desire)$
 8:                 $availableDesires.remove(desire)$
 9:             end if
10:         end if
11:     end for
12:     return $selectedDesires$
13: end procedure
```

The following is an example considering agent $A1$:

Example 5.7: agent $A1$ and Multi-Desire Biased Random Selection.

```
Probabilities held by the beliefs:
    P($B_1$) = 0.4
    P($B_2$) = 0.75
    P(B3) = 0.1

Desires and preconditioning beliefs:
    D1 { B1 }
    D2 { B2 }
    D3 { B3 }

Numeric intervals:
    I1 = [0, 0.4]
    I2 = [0, 0.75]
    I3 = [0, 0.1]

for each desire Di {
    if Di does not conflict with any of the desires selected in this cycle {
        generate random number Ni in the [0,1] interval
        if Ni ∈ Ii then add Di to the list of desires selected in this cycle
    }
}
```

Any subset of the available desires (i.e., $\{\}$, $\{D_1\}$, $\{D_2\}$, $\{D_3\}$, $\{D_1, D_2\}$, $\{D_1, D_3\}$, $\{D_2, D_3\}$, or $\{D_1, D_2, D_3\}$) may be selected in a selection cycle using Multi-Desire Biased Random Selection for agent $A1$ in the presented state, without conflicts.

The advantage that we see in this approach compared to Biased Lottery is that there are no decreased chances of selection for each desire in the cases where there would be normalization, i.e., in situations where the desire list would allow for a sum of precondition probabilities greater than 1, which would in turn lead to decreased individual desire selection probabilities (see Example 5.5).

Multi-Desire Biased Random Selection guarantees that, in the context of the desire selection process itself, non-conflicting desires cannot influence one another with regard to their probability of being selected, as they are handled independently of each other (e.g., full parallelism is possible). On the other hand, adopting multiple desires at once *could* be heavy on resources; it would depend on specific plan requirements for each particular scenario. Nevertheless, we believe that the approach is relevant in the scope of this work.

The following is an example using $MyBayesianAgent$:

Example 5.8: $MyBayesianAgent$ and Multi-Desire Biased Random Selection.

```
initial beliefs = { // no hard evidence available
    A(true = 0.01, false = 0.99),
    B(true = 0.05, false = 0.95),
    C(true = 0.0189, false = 0.9811), // computed
    D(true = 0.057875, false = 0.942125), // computed
    E(true = 0.06150875, false = 0.93849125) // computed
}

Desires and preconditioning beliefs:
    D.false { E.true (0.06150875) }
    B.false { D.true (0.057875) }

Numeric intervals:
    I(D.false) = [0, 0.06150875]
    I(B.false) = [0, 0.057875]
```

Given desires $D.false$ and $B.false$, conditioned on $E.true$ and $D.true$, respectively, each desire is associated with a numeric interval starting at 0 and ending at the numeric value that expresses the probability of that desire's precondition: for $D.false$, the corresponding interval is $[0, 0.06150875]$, and, for $B.false$, the corresponding interval is $[0, 0.057875]$.

- Selection cycle #1:

Desire $D.false$ is under consideration. Number 0.3957 is randomly generated. As it does not belong to $I(D.false)$, the desire is not selected in this cycle.

Desire $B.false$ is under consideration. Number 0.0872 is randomly generated. As it does not belong to $I(B.false)$, the desire is not selected in this cycle.

- Selection cycle #2:

Desire $D.false$ is under consideration. Number 0.0251 is randomly generated. As it does belong to $I(D.false)$, the desire is added to the list of desires to be selected.

Desire $B.false$ is under consideration. Number 0.3812 is randomly generated. As it does belong to $I(B.false)$, the desire is added to the list of desires to be selected.

We traverse the selected desires list and generate intentions for the respective desires using plans $P1$ and $P2$. Both $P1$ and $P2$ fail.

- Selection cycle #3:

Desire $D.false$ is under consideration. Number 0.8609 is randomly generated. As it does not belong to $I(D.false)$, the desire is not selected in this cycle.

Desire $B.false$ is under consideration. Number 0.1304 is randomly generated. As it does not belong to $I(B.false)$, the desire is not selected in this cycle.

- Selection cycle #4:

Desire $D.false$ is under consideration. Number 0.0451 is randomly generated. As it does belong to $I(D.false)$, the desire is added to the list of desires to be selected.

Desire $B.false$ is under consideration. Number 0.4216 is randomly generated. As it does not belong to $I(B.false)$, the desire is not selected in this cycle.

We traverse the selected desires list and generate an intention for $D.false$ using plan $P1$. $P1$ is executed, but establishes that $D = true$, thus failing to fulfill the desire.

```
current beliefs = { // hard evidence that D = true
    A(true = 0.02496760, false = 0.97503240), // computed
    B(true = 0.82107991, false = 0.17892009), // computed
    C(true = 0.03222117, false = 0.96777883), // computed
    D(true = 1.0, false = 0.0),
    E(true = 0.9, false = 0.1) // computed
}

Desires and preconditioning beliefs:
    D.false { E.true (0.9) }
    B.false { D.true (1.0) }

Numeric intervals:
    I(D.false) = [0, 0.9]
    I(B.false) = [0, 1.0]
```

- Selection cycle #5: Desire $D.false$ is under consideration. Number 0.5293 is randomly generated. As it does belong to $I(D.false)$, the desire is added to the list of desires to be selected. Do note that there is hard evidence that $D = true$, i.e., it is known that the current state of the environment contradicts what the agent would like to be true. The belief ascribed as a precondition for the desire does, however, imply that it might be possible to change that (i.e., $P(E = true) = 0.9 > 0$).

Desire $B.false$ is under consideration. Number 0.7038 is randomly generated. As it does belong to $I(B.false)$, the desire is added to the list of desires to be selected.

We traverse the selected desires list and generate intentions for the respective desires using plans $P1$ and $P2$. $P1$ is executed and brings about the belief that $D = false$, thus fulfilling the intention and therefore the desire $D.false$. $P2$ is executed and brings about the belief that $B = false$, thus fulfilling the intention and therefore the desire $B.false$.

```
current beliefs = { // hard evidence that D = false and B = false
    A(true = 0.00909918, false = 0.99090082), // computed
    B(true = 0.0, false = 1.0),
    C(true = 0.01809827, false = 0.98190173), // computed
    D(true = 0.0, false = 1.0),
    E(true = 0.01, false = 0.99) // computed
}
```

It can be argued that postponing the actual desire selection by adding the eligible desires to a list rather than immediately attempting to generate an associated intention is unnecessary, but considering multiple desires for which intentions are expected as a group rather than in isolation may be relevant – e.g., if one single plan happens to be sufficient for aiming to fulfill multiple desires simultaneously. There is no obligation to consider each selected desire in isolation in the actual intention generation process.

5.4 Concrete Example: Watchman agent

5.4.1 Description

In order to illustrate the effects of the desire selection strategies described, we now introduce a working example to show how an agent would react to situations using our proposed algorithms. Our example scenario consists of a watchman agent that is tasked with guarding an installation and reporting anything out of the ordinary. The presence of suspicious people nearby increases its estimate of a security breach. There is an alarm in the installation, that is effective under normal circumstances. However, there are reports of occasional electrical malfunctions in the installation, which may cause the alarm to ring for no reason or not to ring when it is expected to. Moreover, the watchman becomes interested in seeking evidence that there is not an electrical malfunction if it knows that there are suspicious people nearby. If an electrical malfunction is detected, the watchman must report this. The surrounding area is known for intense traffic, and accidents are more common than in most other areas, resulting in noise that is almost always perceived by the agent. However, noise might be caused by trespassers, though that is not very likely. If the watchman finds out that an accident has occurred, this must be reported as well (e.g., so others keep this in mind if they hear noise). In order to patrol the installation, the watchman periodically chooses between the default and an alternate route, and it becomes more inclined to patrol the alternate route as its belief that a security breach is either imminent or already taking place increases, and conversely, the watchman is more inclined to patrol the default route when everything looks calm. The watchman is also expected to keep an eye open for electrical malfunctions and accidents.

Regarding the relationships among the event variables in the network, we note that: *i)* Evidence of the presence of suspicious people nearby increases the probability of a security breach; *ii)* Evidence of the alarm activating increases the probability of a security breach occurring, as well as the probability of there being suspicious people nearby. This is still true if there is also evidence of an electrical malfunction, but the probability increase for both event variable states is smaller. If there is evidence that there is no electrical malfunction (e.g., a notification about maintenance very recently performed), the probability increase is the greatest of the three cases; *iii)* Evidence of noise increases the probability of a security breach. However, this increase is almost nullified upon evidence of an accident, as this network tells us that an accident is a much more probable cause of noise than a security breach, and the impact of a security breach on the probability of noise if we already know of an accident is small; and *iv)* An increase on the probability of a security breach (e.g., through evidence of suspicious people and noise) increases the probability of the alarm activating, even if there is an electrical malfunction, though then the probability increase is smaller.

The Bayesian Network encoding the domain knowledge described in the scenario is represented in Figure 5.1, and it represents the *Watchman* agent's beliefs. Do note that we

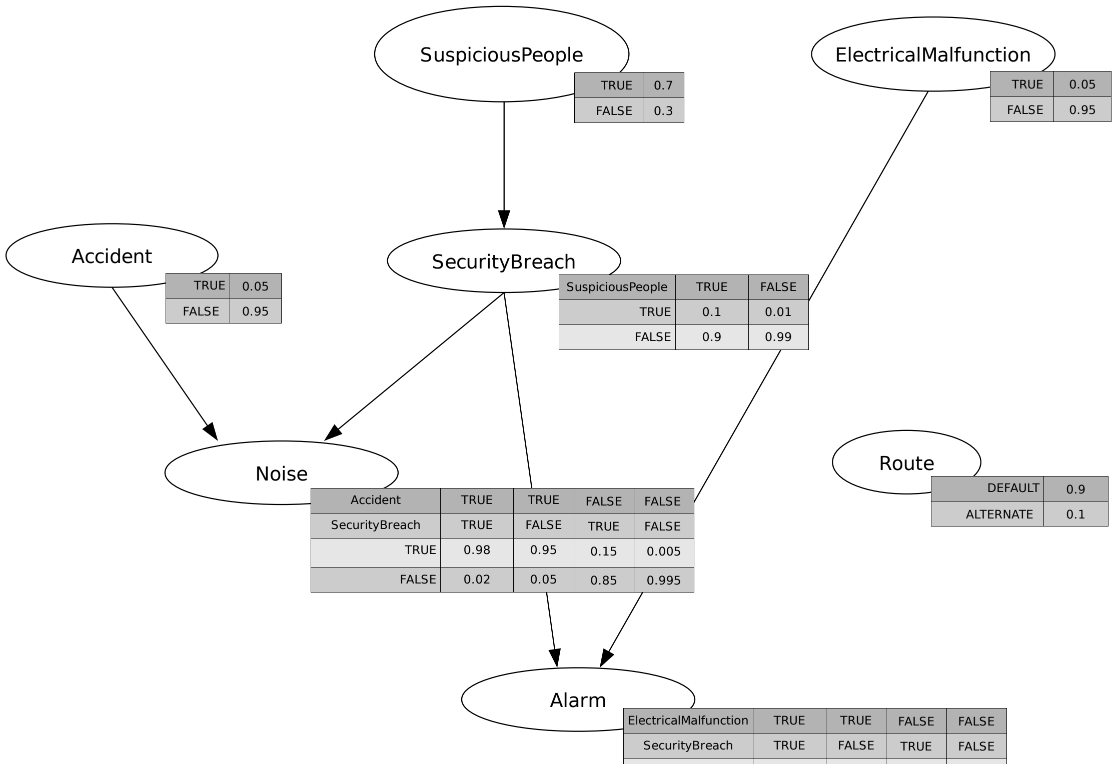


Figure 5.1: Bayesian Network corresponding to the *Watchman* agent's beliefs

do not associate the $Route$ variable with a belief about the environment state, but rather we associate it with an internal belief associated with the agent's currently chosen route. It is not a part of the reasoning surrounding the probability of a security breach or any of the other event variables, and this is the reason we left it disconnected from all the other nodes in the network (i.e., the route is not modeled as a logical cause or consequence of any event therein).

This watchman agent has two mutually exclusive strong desires that are periodically renewed:[2]

- $Route.default(SecurityBreach.false)$; and
- $Route.alternate(SecurityBreach.true)$.

That is, the agent desires to patrol the default route if there has not been a security breach and the alternate one otherwise. It also has the strong desires $ElectricalMalfunction.false(SuspiciousPeople.true)$ – the desire to believe that there is no electrical malfunction at the moment, conditioned on the probability that there are suspicious people nearby – and $Accident.true(Noise.true)$ – the desire to discover that there has been an accident, if noise has been heard.

Note that evidence on event variables $ElectricalMalfunction$ and $Accident$ alone does not affect the state probabilities of event variable $SecurityBreach$. Only if evidence on $Alarm$ is obtained does evidence on $ElectricalMalfunction$ impact event variable $SecurityBreach$ (e.g., from $P(SecurityBreach|ElectricalMalfunction = false) = P(SecurityBreach) = (0.073, 0.927)$[3] to $P(SecurityBreach|Alarm = true, ElectricalMalfunction = false) = (0.8863, 0.1137)$), just as only if evidence on $Noise$ is obtained does evidence on $Accident$ impact $SecurityBreach$ (e.g., from $P(SecurityBreach|Accident = false) = (0.073, 0.927)$ to $P(SecurityBreach|Noise = true, Accident = false) = (0.7026, 0.2974)$). Both are cases of converging connections, from $ElectricalMalfunction$ and $SecurityBreach$ to $Alarm$, and from $Accident$ and $SecurityBreach$ to $Noise$, respectively. Additionally, if the state probabilities of event variable $SecurityBreach$ are updated, as in the cases just described, so are those of $SuspiciousPeople$, if the latter is also uninstantiated.

5.4.2 Desire selection

We now briefly present the result of using each of the four algorithms while working with an initial scenario – where what happens during the execution of each algorithm is not carried over to the next – where there is no hard evidence of any event. Since there is no hard evidence at this point, $P(SuspiciousPeople) = (0.7, 0.3)$ (i.e., $P(SuspiciousPeople = true) = 0.7$ and $P(SuspiciousPeople = false) = 0.3$), and consequently $P(SecurityBreach) = (0.073, 0.927)$; also, $P(Noise) = (0.06241525, 0.93758475)$. Desire $Route.default$ is preconditioned on a belief that $SecurityBreach$ is $false$, which has a 0.927 probability; desire $Route.alternate$ is preconditioned on a belief that $SecurityBreach$ is $true$, which holds a 0.073 probability; desire $ElectricalMalfunction.false$ is preconditioned on a belief that $SuspiciousPeople$ is $true$, which holds a 0.7 probability; and desire $Accident.true$ is preconditioned on a belief that $Noise$ is $true$, which holds a 0.06241525 probability.

First, let us consider threshold-based desire selection, with a threshold of 0.75. This means that only $Route.default(SecurityBreach.false)$ is an eligible desire for selection, since $P(SecurityBreach = false) = 0.927$. In a scenario where there is hard evidence of noise (i.e., $P(Noise = true) = 1$), the probability of suspicious people nearby is increased: $P(SuspiciousPeople = true|Noise = true) = 0.74216638$. However, desire $ElectricalMalfunction.false(SuspiciousPeople.true)$ still fails to satisfy our threshold even so ($0.74216638 < 0.75$). A lower threshold would work in this case,

[2]We denote desires in the form `<desire>(<preconditioning belief>)`, where both elements are described as `<event variable>.<state>`.

[3]The probabilities of the event variable states – *true* and *false*, in this case.

but a precondition's probability might always be smaller than the threshold, if evidence able to increase its probability enough is never obtained – this exemplifies how there may be desires that *are never* selected using this criterion. One might suggest simply lowering the threshold to an extremely low value, but without other criteria we would then simply have a desire selection process that is indifferent to the various probabilities presented.

If we use Probability Ranking desire selection, we get the following ranking:

1. $Route.default(SecurityBreach.false)$: 0.927
2. $ElectricalMalfunction.false(SuspiciousPeople.true)$: 0.7
3. $Route.alternate(SecurityBreach.true)$: 0.073
4. $Accident.true(Noise.true)$: 0.06241525

The agent will desire to patrol the default route, then to establish that there is no electrical malfunction, and finally to patrol the alternate route, in this order, unless a belief update (e.g., evidence that $SecurityBreach = true$) causes the ranking to be modified. Note that although the preconditioning probabilities serve as a criterion for sorting the desires, the probability values by themselves have no impact on how often the desires may be selected, so $Route.alternate(SecurityBreach.true)$ will be promptly selected in the absence of higher-ranked desires regardless of the fact that its precondition holds a low probability.

If we use Biased Lottery, we get the following list of numeric intervals for the desires (the order is irrelevant):

- $Route.default(SecurityBreach.false)$: $[0.0, 0.52598274)$
- $Route.alternate(SecurityBreach.true)$: $[0.52598274, 0.56740317)$
- $ElectricalMalfunction.false(SuspiciousPeople.true)$: $[0.56740317, 0.96458539)$
- $Accident.true(Noise.true)$: $[0.96458539, 1.0]$

The sum of the desires' precondition probabilities is greater than 1, so these values are normalized in the $[0, 1]$ interval and used to generate the intervals. Following the algorithm, a numeric value in the $[0, 1]$ interval is generated, and whichever interval it belongs to determine which desire is selected – if there were not a normalization, it could also tell us that no desire should be selected, by not belonging to any of the intervals. In this example, desire $Route.default(SecurityBreach.false)$ has a 0.52598274 probability of being selected, desire $Route.alternate(SecurityBreach.true)$ has a 0.04142043 probability of being selected, desire $ElectricalMalfunction.false(SuspiciousPeople.true)$ has a 0.39718222 probability of being selected, and desire $Accident.true(Noise.true)$ has a 0.03541461 probability of being selected, each one competing with the others. So, if the randomly generated number is 0.3 (and thus within the first interval), the agent performs a patrol through the default route, or if the random number is 0.55, the patrol is through the alternate route.

If we use Multi-Desire Biased Random Selection, we get the following list of numeric intervals for the desires (in any order):

- $Route.default(SecurityBreach.false)$: $[0.0, 0.927]$

- $Route.alternate(SecurityBreach.true)$: $[0.0, 0.073]$
- $ElectricalMalfunction.false(SuspiciousPeople.true)$: $[0.0, 0.7]$
- $Accident.true(Noise.true)$: $[0.0, 0.06241525]$

For each of the four desires a numeric value in the $[0, 1]$ interval is generated, and if the numeric value belongs to the corresponding desire's numeric interval, the desire is selected. In this example, desires $Route.default(SecurityBreach.false)$, $Route.alternate(SecurityBreach.true)$, $ElectricalMalfunction.false(SuspiciousPeople.true)$ and $Accident.true(Noise.true)$ have, respectively, 0.927, 0.073, 0.7 and 0.06241525 probabilities of being selected when given the chance (i.e., when a random number is generated and checked against the interval designated to the desire). The selection of desires $ElectricalMalfunction.false(SuspiciousPeople.true)$ and $Accident.true(Noise.true)$ is fully independent of the others, while desires $Route.default(SecurityBreach.false)$ and $Route.alternate(SecurityBreach.true)$ conflict with each other, and, in cases where one of them is selected, this prevents the other from getting its chance.

5.4.3 Experimentation

In order to further illustrate agent behavior while using the various presented desire selection algorithms, we implement limited experiments[4].

For our experiments, we create an environment corresponding to the installation, responsible for generating events and providing information on their occurrence to inquiring watchmen (agents). All agents in the experiments are mapped to the same environment, and are thus faced with the same installation events, despite the nondeterministic nature of the latter.

In each cycle, the installation computes the occurrence of a security breach, an electrical malfunction and an accident, each with their own probabilities, which are only visible to the environment. These probabilities do not have an obligation to reflect the probabilities found in the watchman's beliefs, as the latter are but estimates. Nevertheless, conceptually speaking, we assume that, ordinarily, an agent's beliefs have at least *some* have merit to them, and therefore represent information relating to the environment with *some* reliability. Our experiments are limited in that the environment resets the agents every cycle, and these select desires and execute the applicable plans all in one of their own cycles. These plans may or may not be successful in fulfilling the corresponding desires.

The default route, which is the route associated with the belief that there is not a security breach, has a much lower chance of letting the watchman encounter one, but it is also significantly shorter, and the watchman usually chooses to patrol it, under normal circumstances. The alternate route is the opposite: on it, the watchman has a greater chance of detecting a security breach, but its length makes it so that a patrol without detecting a security breach is considered undesirable. Aside from detecting security breaches, the watchman must also beware of electrical malfunctions and accidents, making the appropriate investigations. The data presents the desire selections and their implications, specially in terms of detection counts and distance patrolled per security breach.

Our experiments run over 10000 cycles, with the installation having a 0.005 probability of a security breach along the default route, a 0.1 probability of a security breach along

[4]In the Java programming language, using jSMILE, the SMILE (PITTSBURGH, 2013) library wrapper for Java – SMILE is a library for reasoning in graphical probabilistic models, such as bayesian networks.

the alternate route, a 0.05 probability of an electrical malfunction and a 0.05 probability of an accident. Also, the assigned lengths for the default and alternate routes are 10 and 30, respectively. We present data from Experiments (i) and (ii) in Tables 5.1 and 5.2, where the columns refer to, from left to right: the number of patrols on the default route (*Pat(D)#*), the number of security breaches detected along the default route (*SecBr(D)#*), the number of patrols on the alternate route (*Pat(A)#*), the number of security breaches detected along the alternate route (*SecBr(A)#*), the average distance patrolled per security breach detected (*Dist:SecBr*), the number of electrical malfunction investigations (*Inv(ElMa)#*), the number of electrical malfunction detections (*ElMa#*), the number of accident investigations (*Inv(Acc)#*), and the number of accident detections (*Acc#*). Electrical malfunction and accident investigations refer to the times when the corresponding desires have been selected and are pursued. The rows refer to: the environment, an agent that uses threshold-based selection (*THR*), an agent that uses ProbabilityRanking (*PRK*), the average of five agents that use Biased Lottery (*BLO*) and the average of five agents that use Multi-Desire Biased Random Selection (*MDR*). We use multiple agents for Biased Lottery and Multi-Desire Biased Random Selection due to the variation that results from their nondeterminism.

– *Experiment (i)*: the environment does not provide evidence other than what is necessary for desire satisfaction evaluation – i.e., evidence on event variables $Route$, $ElectricalMalfunction$ and $Alarm$, when the events in question do occur and the agent specifically asks about them (when executing a corresponding plan – this invariably means attempting to obtain evidence, in our experiments). In total, there are 54 security breaches along the default route, 1045 security breaches along the alternate route, 504 electrical malfunctions and 491 accidents. Agent THR puts all its efforts into patrolling the default route, ignoring all the other desires, meaning that it does not ever investigate electrical malfunctions or accidents; also, it presents a comparatively poor distance-to-security-breach ratio. Agent PRK does the same, because each cycle the situation in the environment is renewed (all possible events are computed again) – meaning that any updates to the agents in previous cycles become obsolete and the agents are reset. Agent BLO divides its efforts among the desires following a probability distribution that roughly resembles the sizes of the intervals generated using the desires' precondition probabilities (see Section 5.4.2) – an "ideal" run (one without nondeterministic variations) would have 5260 patrols along the default route (0.526 selection probability), 414 patrols along the alternate route (0.0414 selection probability), 3972 electrical malfunction investigations (0.3972 selection probability) and 354 accident investigations (0.0354 selection probability). Agent MDR covers the possible desires following a probability distribution where only the patrols actually compete with each other (as a result of conflict; we order the list of desires in ascending order of precondition probability), i.e., this watchman does not refrain from investigating electrical malfunctions and accidents because of the patrols, and other than this the precondition probabilities serve as references for the selection rates. Not considering desire conflicts, an "ideal" run for agent MBR would have 9270 patrols along the default route, 730 patrols along the alternate route, 7000 electrical malfunction investigations and 624 accident investigations. Agents BLO and MDR present better (i.e., smaller) distance-per-security-breach ratios than THR and PRK.

– *Experiment (ii)*: the environment always provides evidence of $Alarm = true$ to the agents, in addition to the cases expected in Experiment (i). In total, there are 49 security breaches along the default route, 991 security breaches along the alternate route, 488 electrical malfunctions and 547 accidents. Agent THR puts all its efforts into investigat-

	Pat(D)#	**SecBr(D)#**	Pat(A)#	**SecBr(A)#**	**Dist:SecBr**	Inv(ElMa)#	**ElMa#**	Inv(Acc)#	**Acc#**
ENV	-	**54**	-	**1045**	**-**	-	**504**	-	**491**
THR	10000	**54**	0	**0**	**1851.8519**	0	**0**	0	**0**
PRK	10000	**54**	0	**0**	**1851.8519**	0	**0**	0	**0**
BLO	5274	**32**	422	**42**	**883.7838**	3956	**205**	348	**15**
MDR	8606	**47**	702	**68**	**931.4783**	7009	**351**	630	**29**

Table 5.1: Data gathered in Experiment (i).

ing electrical malfunctions, doing nothing else. Agent PRK does the same. Agent BLO divides its efforts among the desires now giving less attention to the default route, more attention to the alternate route, less attention to investigating electrical malfunctions and more attention to investigating accidents – an ideal run would have 1364 patrols along the default route (0.1364 selection probability), 3552 patrols along the alternate route (0.3552) selection probability, 4333 electrical malfunction investigations (0.4333 selection probability) and 751 accident investigations (0.0751 selection probability). Agent MDR displays the same changes in behavior from Experiment (i) as agent BLO in terms of desire selection rates, except that only the patrols compete with each other. Not considering desire conflicts, an "ideal" run for agent MBR would have 2774 patrols along the default route, 7225 patrols along the alternate route, 8814 electrical malfunction investigations and 1529 accident investigations. Agents THR and PRK do not have distance-per-security-breach ratios in this experiment, since they never detect a security breach (or even make patrols).

	Pat(D)#	**SecBr(D)#**	Pat(A)#	**SecBr(A)#**	**Dist:SecBr**	Inv(ElMa)#	**ElMa#**	Inv(Acc)#	**Acc#**
ENV	-	**49**	-	**991**	**-**	-	**488**	-	**547**
THR	0	**0**	0	**0**	**-**	10000	**488**	0	**0**
PRK	0	**0**	0	**0**	**-**	10000	**488**	0	**0**
BLO	1369	**7**	3538	**348**	**337.5493**	4316	**205**	777	**43**
MDR	2736	**14**	5238	**512**	**350.7605**	8813	**432**	1550	**85**

Table 5.2: Data gathered in Experiment (ii) – $Alarm = true$.

Threshold-based selection is better suited to clinging to desires that are supported by high probabilities at the moment than the other algorithms, since they are more prone to being "distracted" by other desires. Since in our experiments the agent is reset every cycle, Probability Ranking yields the same practical results as threshold-based selection (there is always a desire selected by the latter), instead of giving a chance to desires supported by low probabilities – this would allow for particularly strong cases of said distraction. We can see that Biased Lottery and Multi-Desire Biased Random Selection make the agent spread its efforts among the desires, so they are more varied, less predictable and more inclusive. Neither of these blindly cling to certain desires, and this can be a strength or a weakness depending on the context. Multi-Desire Biased Random Selection is the only algorithm here that usually selects more than one desire per cycle, and in our experiments the agent in question immediately proceeds to deal with them all in the same cycle, taking a more aggressive approach. If resources were taken into account, however, this could be considered inappropriate.

We now discuss the desire selection algorithms presented in this chapter.

5.5 Discussion

Given the four presented algorithms that may be used for desire selection in Bayesian BDI agents, we discuss their characteristics.

From a conservative standpoint, one may argue that Threshold-based selection is sensible as it is, as resources are not used *without justification*. However, we believe that ignoring desires that are probabilistically irrelevant in desire selection is not necessarily a rational choice, since it precludes an agent from exploring the environment.

In Probability Ranking selection, desires that would be ignored by a threshold-based selection do get a chance, though only after the ones that would be accepted by it. However, it might be undesirable to select a desire preconditioned on a belief holding a very low probability just because there is no better alternative.

In Biased Lottery, we rely on nondeterminism, in the computationally feasible form of pseudorandom number generation, to consider all desires while keeping in mind that desires backed by beliefs holding high probabilities should be selected more often than those backed by beliefs holding low probabilities, in proportion to their probabilities. Ideally, the probability of selection of each desire would be the same as the one associated with its precondition, and this is what the pseudorandomness, along with numeric intervals proportional to the probability values involved, aims to emulate. However, in the cases where the sum of the probability values is greater than 1, the competition between the desires in question diminishes the individual probabilities of selection for each desire. Nonetheless, the relative proportions among the probabilities of desire selection are maintained.

In Multi-Desire Biased Random Selection, we also rely on nondeterminism, for the same reason. A key difference is that the number of desires possibly selected is not limited to one, and non-conflicting desires are considered independently of one another.

In both Biased Lottery and Multi-Desire Biased Random Selection, the quality of the pseudorandom number generator does play a role in how well these algorithms emulate the theoretical probability of each desire being selected. Their nondeterministic nature makes it so that agent behavior may not be exactly anticipated by a third party (e.g., another agent) intent on exploiting it. In the case of the watchman agent, such an exploitation could involve forging an accident to drive away suspicion arising from a noise heard by the watchman, for instance. Hiring someone on the inside to plant false evidence of an electrical malfunction would also impact the watchman's beliefs, as a second, albeit more roundabout method of attempting to manipulate the watchman. This is an agent trait that we now describe as *unpredictable proactiveness*: agent behavior at a specific point in time cannot be completely determined by analyzing its beliefs, and is thus resistant to exploitation by a third party.

This concludes our chapter relating to Bayesian BDI Desire Selection. Chapter 6 presents our Final Considerations.

6 FINAL CONSIDERATIONS

Our original goal was to provide planning for the BDI agents that supported uncertainty, given prior knowledge of GRAPHPLAN (BLUM; FURST, 1997) in traditional BDI agents; GRAPHPLAN is a fast planning algorithm[1] that represents the flow of time through a data structure called the *planning graph* – containing proposition levels and action levels – and performs a backward chaining search for an optimal plan. Before we realized that we could not underestimate the relevance of or disregard Bayesian Networks, we investigated two extensions of GRAPHPLAN while aiming to deal with planning supporting uncertainty: PGRAPHPLAN and TGRAPHPLAN (BLUM; LANGFORD, 1998; LITTLE; THIÉBAUX, 2006). The concern with uncertainty eventually led us to Bayesian Networks. However, this soon met with considerable difficulty, as we often came upon literature essentially relying on theoretical formalisms. A primary driving force behind this work was the study of Bayesian Networks, which took most of the invested effort, using knowledge sources deeply reliant on formalisms and a very non-algorithm-driven approach.

Fagundes and Vicari (FAGUNDES; VICARI; COELHO, 2007) presented an initial solution for the desire selection process in their model of Bayesian BDI agents (FAGUNDES; VICARI, 2007), and we presented a limitation and proposed alternatives. It proved to deserve more attention than anticipated, considering that detailed knowledge of Bayesian Networks was not simply assumed, but studied and, thus, presented in this dissertation. Thanks to this, our original goal, Planning for Bayesian BDI Agents, could not be pursued to length.

In Chapter 2, we talked about BDI Agents. We explained the main structure of a BDI agent with regard to its underlying essential components: beliefs, desires and intentions, as well as plans. The execution logic of a traditional BDI agent has also been presented. We pointed out that the occurrence of changes in the environment potentially impacts desires as to their validity – as to whether their selection conditions still hold – and that blindly clinging to always either re-evaluating the desires in this regard or the opposite – ignoring the fact that there have been changes – is hardly an optimal policy. Nevertheless, this is a scenario to be coped with somehow (e.g., a seemingly good idea would involve statistics-based decisions). Those changes might also impact the viability of current plans and even their associated intentions, should there be no possible plans to fulfill them – a scenario fathomable in an optional re-planning phase. Plans meant to satisfy intentions may be generated as needed via specific planning algorithms and/or retrieved from a plan library, which may also contain manually defined plans.

Chapter 3 initially discussed how the awareness of uncertainty in reasoning is relevant

[1]Do note that planning itself is a costly problem (NEBEL, 2000).

in order to broaden the spectrum of treatable problems, for a lack of desirable information is indeed a component of realistic scenarios. Then we presented some pertinent Probability Theory, including conditional probabilitities and Probability Calculus. After this, we covered *Bayes' Rule* – which serves as the foundation for Bayesian Networks, and finally focused on Bayesian Networks themselves. The three connection types found in Bayesian Networks and the *d-separation* property have been presented, as well as other properties and processes relevant to the creation and update of Bayesian Networks.

Chapter 4 presented the concept of Bayesian BDI agents: autonomous agents based on the BDI agent model, but expanded upon it so as to employ Bayesian Networks as the representation mechanism for beliefs, around which desires are defined. It presented a possible execution cycle, as well as its concepts of *beliefs*, *desires* and *intentions*. Furthermore, we proposed the concepts of *strong desires* and *weak desires*, in order to account for both certainty and uncertainty when assessing whether a desire has been satisfied.

Chapter 5 covered approaches to desire selection for Bayesian BDI agents. It explained how a characteristic of a previous method failed to handle all possible desires – to handle the unlikely-albeit-possible ones. In order to work around this shortcoming, we proceeded to propose and detail alternate desire selection methods: *Probability Ranking*, *Biased Lottery* and *Multi-Desire Biased Random Selection*. The first selects desires using a list of desires sorted in decreasing order of the probabilities held by the corresponding preconditions; the second nondeterministically selects desires, one at a time, according to a probability distribution meant to reflect the aforementioned probabilities, except for when normalization is done due to the probability sum being greater than 1; the third nondeterministically selects multiple desires according to those probabilities, and, except when desires conflict, the individual desire selection evaluations are independent of each other. We also presented the *Watchman* agent example, to help illustrate the presented algorithms.

Possible future work aims include the modeling of Bayesian BDI agents focusing on specific scenarios, the evaluation of the presented alghorithms in such scenarios and joint uses of Biased Lottery and Multi-Desire Biased Random Selection while considering possible desire incompatibilities. Scenarios in games, e.g., breaking into a guarded installation – where the guards are agents – present an avenue for applied experiments. Furthermore, *plans* for Bayesian BDI agents seem worth investigating, e.g., plans in practical modeled examples, as well as considering *utility* and *cost* in desire selection.

α1 INTRODUÇÃO

A partir deste capítulo segue uma versão reduzida, em Português do Brasil, da presente dissertação de mestrado.

Engenharia de software tradicional não possui um foco inerente em sistemas capazes de ações autônomas flexíveis voltadas a atingirem certos objetivos. Por outro lado, a área de agentes autônomos na IA é composta por entidades que sentem o ambiente em que estão situadas e são capazes de ações autônomas naquele ambiente. Um modelo de agente autônomo que tem sido foco de estudo é o modelo de agente BDI.

O modelo de agente BDI (RAO; GEORGEFF, 1995) gira em torno de três conceitos: crenças, desejos e intenções. Crenças correspondem ao que o agente acredita sobre o ambiente ("o mundo"), desejos correspondem ao que o agente *gostaria* que fosse verdade, e intenções são aqueles desejos com os quais o agente se comprometeu.

Um agente BDI tradicional não representa incerteza em suas crenças. Um agente BDI pode usar um conjunto fechado de literais para representar suas crenças. Uma crença é geralmente associada a uma verdade lógica, não deixando margem mesmo para estados *desconhecidos*. Conhecimento incerto que suporte a noção de múltiplos variáveis níveis de certeza / confiabilidade – algo além da ideia de um simples estado desconhecido – pode ser representado através de um modelo probabilístico gráfico conhecido como uma Rede Bayesiana.

Redes Bayesianas (PEARL, 1988) são uma maneira popular de representar informação incerta probabilisticamente. Elas são grafos acíclicos dirigidos, cujos nodos representam variáveis de evento associadas a dois ou mais possíveis estados. Cada estado possui uma probabilidade explícita de ocorrer.

Dada a falta de suporte à incerteza no modelo de agente BDI e o poder de representação de Redes Bayesianas, um modelo de agente estendido que integra BDI e Redes Bayesianas é provido por Fagundes e Vicari (FAGUNDES; VICARI, 2007). Esse modelo de agente baseado em BDI não mais depende de literais, mas sim de uma Rede Bayesiana. Isto tem um impacto no processo de raciocínio: processos de agente BDI "regular" para seleção de desejos, especificação de ações e planejamento deixam de ser apropriados.

O processo de seleção de desejos é inerentemente suscetível a acabar escolhendo desejos que acabem mostrando-se impossíveis de satisfazer em um dado estado do mundo, independentemente do quão promissores possam parecer, i.e., apesar de qualquer estimativa satisfatória baseada na Rede Bayesiana. Além disso, mesmo que um agente considere as chances de um desejo ser satisfeito pequenas demais, não há garantias de que o último seja irrelevante, sob incerteza.

Esta dissertação de mestrado investiga o funcionamento de agentes autônomos que fazem uso de Redes Bayesianas para representação de crenças incertas, e oferece abordagens de seleção de desejos alternativas àquela proposta por Fagundes et al. (FAGUNDES;

VICARI; COELHO, 2007), de modo a não desconsiderar desejos pré-condicionados por crenças associadas a baixas probabilidades.

α1.1 Objetivos e estrutura

Nossos objetivos específicos são:

1. apresentar o que é um agente BDI;
2. apresentar limitações de um agente BDI clássico;
3. apresentar Teoria da Probabilidade e Redes Bayesianas enquanto buscamos evitar depender de formalismo;
4. apresentar a união do paradigma de agente BDI e Redes Bayesianas como uma solução para superar certas limitações do agente BDI clássico;
5. delinear certas limitações de uma solução prévia e oferecer mecanismos de seleção de desejos alternativos.

Esta dissertação está organizada conforme a seguir: o Capítulo α2 foca no modelo de agente BDI, o Capítulo α3 explica Redes Bayesianas, o Capítulo α4 revisa a integração entre o modelo de agente BDI e Redes Bayesianas, e o Capítulo α5 apresenta considerações quanto à seleção de desejos para este modelo de agente estendido.

α2 AGENTES BDI

Jennings (JENNINGS, 2000) define um agente autônomo como um sistema computacional encapsulado *situado* em um ambiente e capaz de ações autônomas *flexíveis* neste ambiente, a fim de atingir certos objetivos. O agente deve se adaptar a um ambiente dinâmico, enquanto busca realizar seus objetivos. Este capítulo se concentra nos Objetivos 1 e 2 (os objetivos estão listados na Seção α1.1), i.e., apresentar o que é um agente BDI e algumas de suas limitações.

Um modelo de agente autônomo que tem recebido atenção como objeto de pesquisa e desenvolvimento apresenta crenças, desejos e intenções, sendo chamado de modelo de agente BDI (*Belief-Desire-Intention*) (RAO; GEORGEFF, 1995). O modelo de agente BDI é baseado na teoria de raciocínio prático de Bratman (BRATMAN; ISRAEL; POLLACK, 1988), que é relacionada a perseguir objetivos sob restrições de recursos, estabelecendo os meios – uma série ordenada de passos lógicos – para um fim. Como qualquer agente, ele pode ser descrito como possuindo sensores e atuadores, os quais percebem e agem sobre o ambiente, respectivamente.

Crenças contém uma representação, interna ao agente, de elementos do ambiente considerados relevantes para o raciocínio do agente. O estado das crenças do agente pode conter ou menos informação que o estado atual do ambiente (e.g., devido a limitações dos sensores), ou mais (e.g., se o agente realiza processamento adicional com base em informações captadas pelos sensores). Desejos são os objetivos que o agente gostaria de atingir (i.e., eles podem ser considerados a *motivação* do agente (RAO; GEORGEFF, 1995)). Intenções são aqueles desejos com cuja satisfação o agente se comprometeu, através de planos. Planos são sequências de ações voltadas a realizar intenções. Agentes resistem abandonar suas intenções, e caso um plano falhe, pode ser que eles escolham realizar replanejamento, apesar do custo geralmente não-insignificante disto.

Um agente BDI seleciona desejos através de um processo que considera suas viabilidades atuais e a ausência de conflitos com intenções existentes. Desejos comumente possuem crenças pré-condicionantes que indicam se eles devem ser selecionados pelo agente, como uma questão de avaliação lógica (MÓRA et al., 1999). Todavia, mudanças podem ocorrer no ambiente durante a execução deste processo. Tais mudanças podem ou não ser relevantes com relação a quais desejos deveriam ser selecionados, se algum; crenças que pré-condicionam desejos podem ou não ter sido afetadas. Não há curso de ação “correto”: por um lado, recomeçar ou mesmo reverter o processo é uma perda de tempo se os desejos poderiam ser escolhidos apesar das crenças desatualizadas (que acabaram de ser modificadas) – i.e., se os desejos continuam de fato válidos; por outro lado, prosseguir com a seleção de desejos dependentes de crenças que agora contradizem o ambiente é também um prospecto indesejável. Pode ser argumentado que prosseguir com o planejamento no último cenário é especialmente computacionalmente dispendioso, mas se a frequência na qual isto acontece for suficientemente baixa, ainda é uma decisão justificada. Balancear o uso destas duas opções baseando-se na probabilidade estatística de cada uma delas ser apropriada, assim como em seus respectivos custos computacionais medidos, parece ser uma resposta sensata para este dilema.

Mudanças no ambiente também ocorrem durante a obtenção ou execução de um plano, possivelmente comprometendo sua viabilidade (no caso das mudanças afetarem pré-condições) e, se não há plano alternativo, a viabilidade das intenções associadas. Adicionalmente, tais mudanças podem fazer com que desejos de maior prioridade tornem-se viáveis, o que pode significar que manter as intenções correntes (i.e., manter o comprometimento) é considerado indesejável em contraste ao comportamento esperado do agente em um certo cenário. Entretanto, realizar reconsideração de intenções possui um custo computacional significativo, e uma mudança no ambiente não necessariamente tem um impacto nas intenções do agente. Dado que ambos extremos têm o potencial para gerar resultados condenáveis, um equilíbrio entre reconsideração e comprometimento (i.e., um equilíbrio no quão frequentemente o agente reconsidera suas intenções) deve ser estabelecido (KINNY; GEORGEFF, 1991).

Uma abordagem possível para reconsideração de intenções é fazer uso de *gatilhos* de reconsideração, contendo condições que indiquem que os planos atualmente adotados foram comprometidos ou que objetivos (desejos) mais importantes tornaram-se possíveis (MENEGUZZI et al., 2007), desta forma incumbindo o projetista do agente com definir a relevância da reconsideração caso-a-caso. Mover "inteligência" de um agente autônomo para seu projetista humano é de maneira geral visto como uma prática com a qual deve-se ser cauteloso, entretanto, devido a um número possivelmente intratável de cenários a serem levados em conta, e, se for feito, deve ser feito com parcimônia.

α2.1 Arquitetura

A estrutura do *Procedural Reasoning System* (*PRS*; Sistema de Raciocínio Procedural) (RAO; GEORGEFF, 1995) tem estruturas de dados dinâmicas correspondentes às crenças, desejos e intenções do agente. Além disso, outra estrutura de dados presente é uma fila de eventos. Todos eventos (e.g., mudanças no ambiente) são adicionados à fila para serem processados posteriormente; sem alguma espécie de controle (neste caso, a fila) haveria o risco da cpu não dar conta de eventos simultâneos excessivos.

Algoritmo 1 Interpretador de agente BDI

```
procedure INTERPRETADOR-BDI
    inicializarEstado();
    loop
        opções ← geradorDeOpções(filaDeEventos);
        opçõesSelecionadas ← deliberar(opções);
        atualizarIntenções(opçõesSelecionadas);
        executar();
        obterNovosEventosExternos();
        abandonarComportamentosBemSucedidos();
        abandonarComportamentosImpossíveis();
    end loop
end procedure
```

Pseudocódigo representando um ciclo de execução de um agente BDI do tipo PRS é apresentado no Algoritmo 1. Nesta arquitetura, em cada ciclo de execução o que um agente BDI inicialmente faz é coletar eventos da fila e obter uma lista de desejos (opções) (Algoritmo 1, Linha 4); isto é importante porque um dado desejo pode não ser aplicável, dadas as crenças atuais. Logo após, o agente seleciona das opções um subconjunto (Linha 5), a partir do qual ele gera intenções (i.e., o agente se compromete com os desejos selecionados) (Linha 6). O agente executa a proxima ação associada a uma intenção (Linha 7). Então, quaisquer novos eventos externos são obtidos pelos sensores e adicionados à fila de eventos (Linha 8). No final do ciclo, o agente remove desejos e intenções satisfeitos (i.e., desejos que foram atingidos e intenções que foram realizadas) de suas respectivas estruturas de dados (Linha 9), assim como os impossíveis e inviáveis (Linha 10).

Crenças referem-se somente ao tempo atual e podem ser representadas simplesmente como literais *ground*. Os conjuntos de passos para atingir possíveis estados futuros são representados por *planos*, os quais são associados a uma condição de disparo e uma pré-condição. Tais planos podem ou ser providos *a priori* numa especificação feita por um humano ou gerados algoritmicamente (i.e., sem intervenção humana). Adicionalmente, planos podem ser armazenados e obtidos de uma *biblioteca de planos*.

O Capítulo $\alpha 3$ foca em Redes Bayesianas, incluindo Teoria da Probabilidade que as suporta.

α3 REDES BAYESIANAS

α3.1 Incerteza

Incerteza é uma característica encontrada em muitos domínios, no mundo real e em mundos abstratos. Este capítulo lida com o Objetivo 3 (os objetivos estão listados na Seção $\alpha 1.1$), i.e., apresentar teoria pertinente de Probabilidade e Redes Bayesianas de um ponto de vista mais interessado em ser didático do que em empregar formalismos.

Abordagens tradicionais de lógica de primeira ordem para representação do conhecimento são tidas como insuficientes para representar certos domínios onde há incerteza na validade das sentenças ao longo do tempo (RUSSEL; NORVIG, 1994; JENSEN; NIELSEN, 2007). Razões para isto incluem:

- **Preguiça**: o alto custo de representar exaustivamente todas possíveis combinações de valores-verdade usando regras de lógica;

- **Ignorância teórica**: a falta de uma teoria completa do domínio em questão (e.g., diagnóstico médico);

- **Ignorância prática**: a potencial impossibilidade ou inviabilidade de realizar todos os testes necessários para garantir a verdade completa para certas sentenças (e.g., (supõe-se que) não há quantidade de exames de sangue suficiente para levar a um diagnóstico perfeito; todos os equipamentos necessários para um certo conjunto de testes podem não estar disponíveis sob certas restrições temporais).

A verdade é que as pessoas comumente trabalham com conhecimento incompleto e tomam decisões baseadas em suposições sobre fatos desconhecidos. Este conhecimento é composto pelo que é *conhecido*, pelo que é *desconhecido* e *estimativas* baseadas em relacionamentos entre elementos do mundo. Por exemplo, este tipo de raciocínio é realizado por humanos: “Eu tentei ligar meu carro e não funcionou. Eu vejo duas causas possíveis: falta de gasolina ou velas de ignição sujas. Além disso, se uma delas aconteceu, é improvável que a outra também. Eu devo procurar por evidência de uma delas.” Pearl (PEARL, 1988) provê um formalismo para representar conhecimento parcial baseado nos relacionamentos causais entre elementos do mundo, usando Teoria da Probabilidade para representar como conhecimento acerca de um elemento no mundo influencia a certeza sobre outros relacionados a ele. Aqui, relacionamentos entre elementos são representados em uma rede, e probabilidades entre elementos relacionados são calculadas usando a *Regra de Bayes*, com o formalismo resultante sendo denominado *Rede Bayesiana*. Uma Rede Bayesiana é um tipo de rede causal que permite a especificação de conhecimento

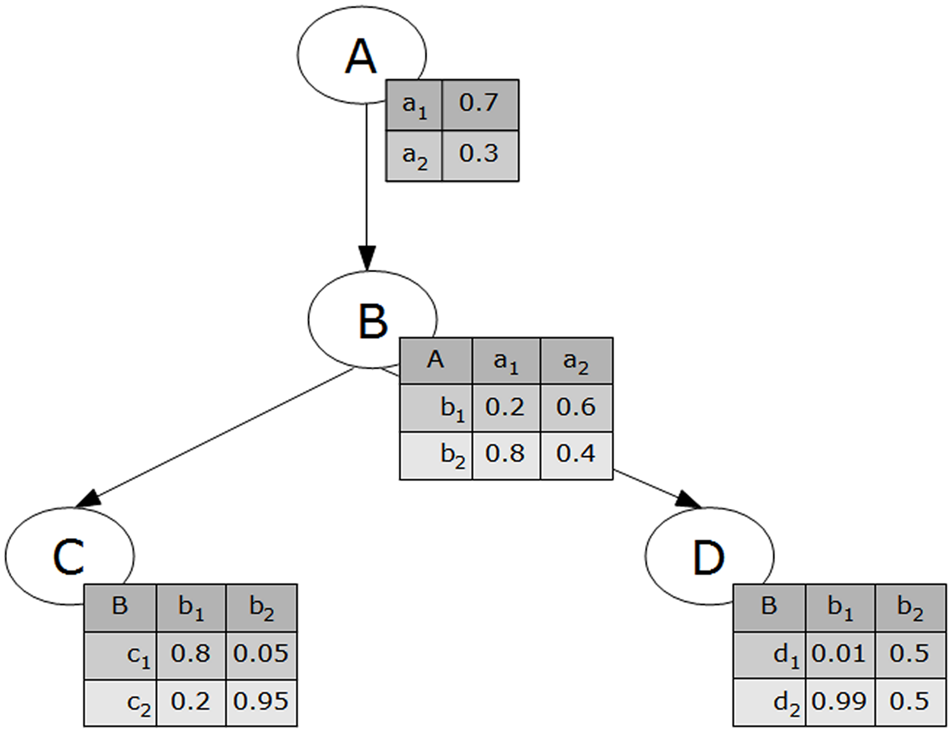


Figura α3.1: Exemplo simples de Rede Bayesiana

onde partes dele estão condicionadas a outras (e.g., relações de causa e consequência, doenças e sintomas), suportando a atualização de probabilidades quando nova informação (i.e., *evidência*) é obtida.

A Figura α3.1 mostra um exemplo de Rede Bayesiana. São dadas 4 variáveis de eventos: A, B, C e D, cada um com dois estados. Cada nodo de variável possui uma tabela com probabilidades associada. A tabela para A nos dá as probabilidades de cada um de seus estados, a tabela para B nós dá as probabilidades de seus estados dado cada um dos estados de A, a tabela para C nos dá as probabilidades de seus estados dado cada um dos estados de B e a tabela para D nos dá as probabilidades para seus estados dado cada um dos estados de B. Por exemplo, a probabilidade que A está no estado a_1 é de 70% e a probabilidade que B está no estado b_2, já sabendo que A está no estado a_1, é de 80%.

A fim de entender como Redes Bayesianas são utilizadas para representar conhecimento incompleto, é importante compreender Teoria da Probabilidade. Consequentemente, a partir da próxima seção nós apresentaremos Teoria da Probabilidade pertinente necessária para compreender Redes Bayesianas, presente em (JENSEN; NIELSEN, 2007).

α3.2 Teoria da Probabilidade

Teoria da Probabilidade é o alicerce para matematicamente sondar o incerto. Ela permite fazer estimativas sobre probabilidades. Agora, apresentamos alguns fundamentos.

α3.2.1 Fundamentos

Quando alguém deseja explorar a natureza de um problema que é repleto de incerteza quanto a seus resultados teoricamente possíveis, esta pessoa pode recorrer à realização de *experimentos*. Um experimento, que pode ser visto como um conjunto de observações voltado à investigação de relacionamentos causais entre variáveis, resulta em uma de múltiplas possibilidades, cada uma com uma probabilidade de ocorrer. O conjunto de todos os possíveis resultados que um experimento gera é chamado de *espaço amostral* do experimento. Um subconjunto do espaço amostral é chamado de um *evento*. É dito que um evento A é *verdadeiro* para um experimento se o resultado é um elemento de A.

Exemplo α3.1 Jogar um dado resulta no espaço amostral $S = \{ 1, 2, 3, 4, 5, 6 \}$, onde cada número possui uma probabilidade de $\frac{1}{6}$ de ser o valor sorteado. O evento que jogando um dado de seis lados obtém-se um número maior que 4 corresponde ao subconjunto $\{ 5, 6 \} \subseteq \{ 1, 2, 3, 4, 5, 6 \}$.

Andrei Kolgomorov mostra como construir o resto da Teoria da Probabilidade a partir de três axiomas, nomeados *Axiomas de Kolgomorov*. As probabilidades para cada evento $A \subseteq S$ devem obedecer os Axiomas de Kolgomorov, apresentados a seguir:

Axioma 1 $P(S) = 1$.

O evento S relativo a obtermos um resultado dentro do espaço amostral é garantido, e consequentemente atribuído a probabilidade 1.

Axioma 2 $\forall\ A \subseteq S$, $P(A) \geq 0$.

Qualquer evento A deve possuir uma probabilidade não-negativa.

Axioma 3 *Se $A \subseteq S$, $B \subseteq S$ e $A \cap B = \oslash$, então $P(A \cup B) = P(A) + P(B)$.*

Se dois eventos A e B são disjuntos, então a probabilidade do evento combinado (i.e., que *pelo menos um* dos eventos seja verdadeiro) é a soma das probabilidades para os dois eventos individuais.

Alternativamente, se A e B não são disjuntos, então

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Notação: Se um evento A contém apenas um resultado a, escrevemos $P(a)$ ao invés de $P(\{a\})$.

α3.2.2 Probabilidades condicionais

Uma sentença sobre a probabilidade é sempre condicionado no que mais é conhecido. Por exemplo, no exemplo do dado, nós implicitamente supomos que o dado é justo (i.e., não é viciado); a probabilidade é condicionada a isto.

Notação: $P(A|B) = p$ significa *"Dado o evento B, a probabilidade do evento A é p."* Se B é verdadeiro, e *todo o resto é irrelevante para A*, então a probabilidade de A é p.

Sejam A e B subconjuntos de S. Se nós sabemos que o evento B ocorre, então todos possíveis resultados são elementos de B, e todos os resultados para os quais A pode ser verdadeiro são $A \cap B$. Para dois eventos A e B, com $P(B) > 0$, a probabilidade condicional para A dado B é:

Definição α3.1: probabilidade condicional. [1]

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Probabilidades condicionais não estão restritas a apenas um evento.

Definição α3.2: probabilidade condicional – versão geral. A definição geral de probabilidade condicional é

$$P(A|B \cap C) = \frac{P(A \cap B \cap C)}{P(B \cap C)}.$$

α3.2.3 Cálculo Probabilístico

Se nós sabemos a probabilidade de A dado B e a probabilidade de B, podemos calcular a probabilidade de ver ambos A e B. A Equação α3.1 pode ser reescrita de modo a obter a *regra fundamental* para o Cálculo Probabilístico:

Definição α3.3: regra fundamental. Dados os eventos A e B,

$$P(A|B)P(B) = P(A \cap B).$$

A probabilidade de observarmos ambos eventos A e B pode ser obtida multiplicando-se a probabilidade de A dado B pela probabilidade de B.

A regra fundamental pode ser condicionada a outro evento C, como a seguir:

Definição α3.4: regra fundamental – versão geral.

$$P(A|B \cap C)P(B|C) = P(A \cap B|C).$$

Dado que $P(A \cap B) = P(B \cap A)$ (e $P(A \cap B|C) = P(B \cap A|C)$), nós obtemos

$$\begin{aligned} P(A \cap B) &= P(B \cap A) \\ P(A|B)P(B) &= P(B|A)P(A) \\ \frac{P(A|B)P(B)}{P(B)} &= \frac{P(B|A)P(A)}{P(B)}, \end{aligned}$$ que leva à *Regra de Bayes* :

Definição α3.5: Regra de Bayes. Dados eventos A e B,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

A Regra de Bayes torna possível atualizar crenças sobre um evento A quando obtivermos informação sobre outro evento B. Desta forma, $P(A)$ é geralmente chamada de

[1] As definições neste capítulo foram extraídas de (JENSEN; NIELSEN, 2007).

probabilidade a priori de A, enquanto $P(A|B)$ é chamada de *probabilidade posterior* de A dado B.

Há também uma versão geral da Regra de Bayes:

Definição α3.6: Regra de Bayes – versão geral. Regra de Bayes em um contexto C:

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}$$

α3.2.4 Independência entre eventos e independência condicional entre eventos

Informação sobre um evento B pode não ter impacto na nossa crença sobre a ocorrência de outro evento A. Neste caso, eles são considerados *independentes*.

Definição α3.7: independência entre eventos. Quaisquer eventos A e B são independentes *se*

$$P(A|B) = P(A).$$

Esta noção é simétrica: se A é independente de B, então B também é independente de A (i.e., $(P(A|B) = P(A)) \leftrightarrow (P(B|A) = P(B))$).

Definição α3.8: regra fundamental para eventos independentes. Quando dois eventos são independentes, a regra fundamental pode ser reescrita como

$$P(A \cap B) = P(A|B)P(B) = P(A) \cdot P(B).$$

Este conceito também existe quando condiciona-se por múltiplos eventos. Se informação sobre o evento B não altera nossa crença sobre o evento A quando nós já conhecemos o evento C, então dizemos que A e B são *condicionalmente independentes* dado C.

Definição α3.9: independência condicional entre eventos. Dois eventos são ditos condicionalmente independentes *se*

$$P(A \cap B|C) = P(A|C) \cdot P(B|C).$$

Quando dois eventos são (incondicionalmente) independentes, isto significa que $C = \oslash$ (i.e., $P(A|\oslash) = P(A)$ e $P(A \cap B|\oslash) = P(A \cap B)$).

Definição α3.10: regra fundamental para eventos condicionalmente independentes. Se quaisquer eventos A e B são independentes, então

$$P(A|B \cap C) = P(A|C).$$

α3.2.5 Cálculo Probabilístico para variáveis

Até agora, consideramos eventos simples e resultados em um único espaço amostral. A partir de agora, consideraremos coleções de espaços amostrais, estes também chamados *variáveis*. Uma variável pode ser considerada um experimento, e para resultado deste experimento a variável possui um *estado* correspondente.

Notação: o conjunto de estados associado a uma variável A é expresso como $sp(A) = (a_1, a_2, \ldots, a_n)$. Estes estados devem ser *mutuamente exclusivos* e *exaustivos*. Usamos letras maiúsculas para variáveis e letras minúsculas para estados. No exemplo do dado, com a variável D representando o resultado do jogar de dados, $sp(D) = (1, 2, 3, 4, 5, 6)$.

Para uma variável A com estados $a_1, \ldots, a_n$, expressamos incerteza sobre seu estado através de uma distribuição de probabilidade $P(A)$ sobre esses estados:

$$P(A) = (x_1, \ldots, x_n); \qquad x_i \geq 0; \qquad \sum_{i=1}^{n} x_i = x_1 + \cdots + x_n = 1,$$

onde x_i é a probabilidade de A encontrar-se no estado a_i. A distribuição de probabilidade é dita *uniforme* se as probabilidades para todos estados são iguais.

Notação: a probabilidade de A encontrar-se no estado a_i é denotada por $P(A = a_i)$, e denotada por $P(a_i)$ se a variável for óbvia pelo contexto.

Probabilidades condicionais também se aplicam a variáveis: se $sp(A) = (a_1, \ldots, a_n)$ e $sp(B) = (b_1, \ldots, b_m)$, então $P(A|B)$ contém $n{\cdot}m$ probabilidades condicionais $P(a_i|b_j)$. Este conjunto de probabilidades geralmente é representado em uma tabela $n \times m$, com uma probabilidade para cada combinação dos estados das variáveis envolvidas. Esta tabela é chamada *Tabela de Probabilidades Condicionais*. Adicionalmente, o Axioma 1 (vide Seção α3.2) nos diz que a soma das probabilidades sobre A deve ser 1 para cada estado de B, como mostrado na equação a seguir. A Tabela α3.1 (JENSEN; NIELSEN, 2007) é uma possível tabela de probabilidades condicionais para $P(A|B)$.

$$\sum_{i=1}^{n} P(A = a_i | B = b_j) = 1 \text{ para cada } b_j.$$

	b_1	b_2	b_3
a_1	0.4	0.3	0.6
a_2	0.6	0.7	0.4

Tabela α3.1: Exemplo de uma tabela de probabilidades condicionais $P(A|B)$

A probabilidade de observarmos resultados conjuntos para experimentos distintos pode ser expressa pela *probabilidade conjunta* de duas ou mais variáveis: para cada configuração (a_i, b_j) de variáveis A e B, $P(A, B)$ especifica a probabilidade de observarmos ambos $A = a_i$ e $B = b_j$. Assim como $P(A|B)$, $P(A, B)$ é representada em uma tabela $n \times m$. Esta é chamada *Tabela de Probabilidades Conjuntas*. Já que os espaços de estados de A e B são mutuamente exclusivos e exaustivos, todas combinações de seus estados são também mutuamente exclusivas e exaustivas – como mostrado na equação a seguir, e elas podem ser consideradas um espaço amostral. A Tabela α3.2 (JENSEN; NIELSEN, 2007) é uma tabela de probabilidades conjuntas possível para $P(A, B)$.

$$P(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(A = a_i, B = b_j) = 1.$$

Se a soma das probabilidades dos estados das variáveis que constituem um dado espaço amostral em algum momento for diferente de 1 como resultado de um cálculo, as probabilidades devem ser *normalizadas*.

	b_1	b_2	b_3
a_1	0.16	0.12	0.12
a_2	0.24	0.28	0.08

Tabela α3.2: Exemplo de uma tabela de probabilidades conjuntas $P(A, B)$

Aplicando a regra fundamental (Definição α3.3) a cada uma das $n{\cdot}m$ (a_i, b_j) variáveis, obtemos

$$P(a_i|b_j)P(b_j) = P(a_i, b_j).$$

Se $P(B) = (0.4, 0.4, 0.2)$, a Tabela α3.2 é o resultado de aplicar a regra fundamental à Tabela α3.1.

Definição α3.11: a regra fundamental para variáveis. A regra fundamental para variáveis é

$$P(A, B) = P(A|B)P(B),$$

e condicionada a uma outra variável C,

$$P(A, B|C) = P(A|B, C)P(B|C).$$

A partir de uma tabela de probabilidades conjuntas $P(A, B)$, a distribuição de probabilidades $P(A)$ pode ser calculada considerando-se os resultados de B que podem ocorrer simultaneamente com cada estado a_i de A. Para cada estado a_i há m resultados mutuamente exclusivos $(a_i, b_1), \ldots, (a_i, b_m)$. Assim, pelo Axioma 3 (vide Seçãoα3.2), obtemos a *marginalização da variável* B de $P(A, B)$.

Definição α3.12: marginalização de variável. A variável de evento B é dita marginalizada de $P(A, B)$ em

$$P(a_i) = \sum_{j=1}^{m} P(a_i, b_j), \text{ onde } P(A) = \sum_{i=1}^{n} P(a_i).$$

Notação: $P(A) = \sum_B P(A, B)$ significa que a variável B é marginalizada de $P(A, B)$, resultando em $P(A)$.

Agora procedemos para seções mais especificamente relacionadas ao aspecto causal de Redes Bayesianas.

α3.3 Tipos de conexão e d-separação

Tabelas de probabilidades crescem exponencialmente com o número de variáveis. A principal vantagem de Redes Bayesianas sobre tabelas exaustivas – tabelas de probabilidades conjuntas completas – é que elas não requerem que todas as variáveis na rede sejam levadas em conta simultaneamente; pode haver variáveis independentes entre si, e estas não aumentam o custo computacional como seria o caso se elas causassem impacto nas probabilidades associadas da outra variável. A propriedade de Redes Bayesianas que nos diz se uma variável é independente de outra é chamada *d-separação*.

Quando há evidencia de que uma determinada variável está em um certo estado, é dito que a variável está *instanciada*. Evidência deste tipo é chamada de *evidência forte*. Por outro lado, se algo é afirmado sobre uma variável com base em dependências ao invés de conhecimento explícito, a evidência sobre a variável é chamada de *evidência fraca*.

A topologia de redes causais é constituída de instâncias de três tipos de conexão: serial, divergente e convergente. Cada tipo de conexão implica em um raciocínio específico quanto às variáveis envolvidas serem *d-separadas* ou *d-conectadas* (o que dizemos de variáveis que não são d-separadas).

α3.3.1 Serial

Em uma conexão serial, se não temos evidência forte sobre uma variável, evidência sobre seu pai/filho passa através dela, afetando crenças sobre este e seu filho/pai não-instanciado. Imagine três variáveis, A, B e C, tal que A está conectado a B e B a C, como na Figura α3.2 (JENSEN; NIELSEN, 2007).

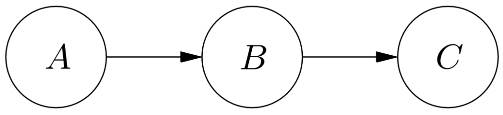


Figura α3.2: Conexão serial. Se B é instanciado, A e C são *d-separados*.

α3.3.2 Divergente

Em uma conexão divergente, uma variável é conectada a duas ou mais variáveis. Se não temos evidência forte sobre o pai, evidência sobre um dos filhos afeta nossas crenças sobre os outros filhos não-instanciados. Imagine cinco variáveis, A, B, C, D e E, onde A está conectado a todas as outras, como na Figura α3.3 (JENSEN; NIELSEN, 2007). B, C, D e E podem transmitir evidência entre si através de A a não ser que A esteja instanciado, "bloqueando a comunicação" entre seus filhos. $B, \ldots, E$ são ditos *d-separados* se A está instanciado e *d-conectados* caso contrário.

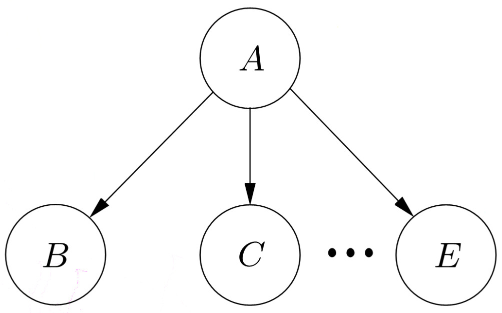


Figura α3.3: Conexão divergente. Se A é instanciado, seus filhos são *d-separados*.

α3.3.3 Convergente

Em uma conexão convergente, duas ou mais variáveis, não-conectadas entre si, são conectadas a uma outra (seu filho), que pode ter descendentes. Se não temos evidência forte sobre o filho ou um de seus descendentes, evidência sobre um pai não influencia nossas crenças sobre o outro. Imagine cinco variáveis, A, B, C, D e E, onde B, C, D e E estão conectados a A, mas não uns aos outros, como na Figura α3.4 (JENSEN; NIELSEN, 2007). A Figura α3.5 (JENSEN; NIELSEN, 2007) mostra duas outras amostras de modelos causais que usam uma conexão convergente. Somente se A ou um de seus descendentes for instanciado os pais de A – B, C, ... e E na Figura α3.4 e B e C na Figura α3.5 – poderão transmitir evidência entre si através de A. Se A e seus descendentes não estão instanciados, os pais de A são ditos *d-separados*. Caso contrário, eles estão *d-conectados*.

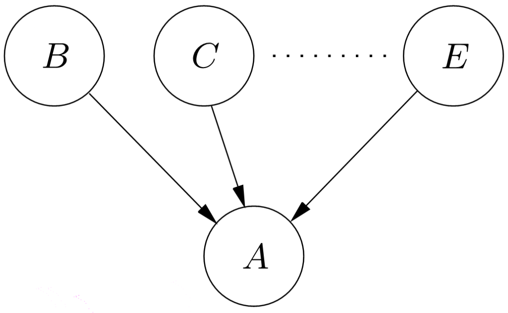


Figura α3.4: Conexão convergente. Se A não está instanciado, nem nenhum de seus descendentes, seus pais estão *d-separados*.

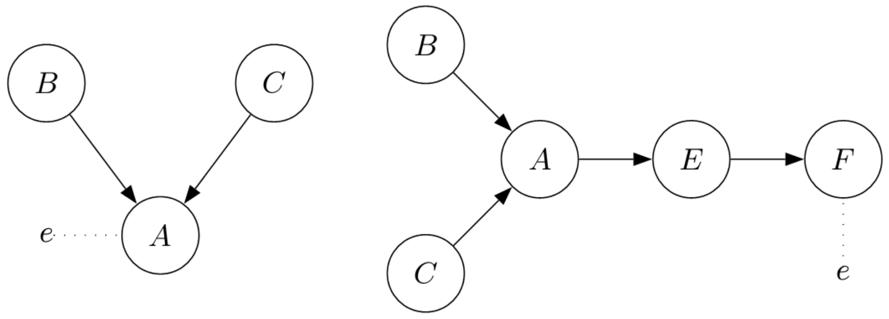


Figura α3.5: Conexões convergentes. Em ambos grafos, dada evidência forte denotada por $e \cdots$, B e C são *d-conectados*.

Definição α3.13: d-separação. *Duas variáveis A e B são d-separadas se para todos os caminhos entre A e B existe uma variável intermediária V, além de A e B, tal que ou*

- a conexão é serial ou divergente e V está instanciado

ou

- a conexão é convergente, e nem V, nem qualquer um dos descendentes de V recebeu evidência.

α3.4 Sumário

Segue um sumário de fórmulas relacionadas a Redes Bayesianas, para fácil referência:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$
$$P(A|B \cap C) = \frac{P(A \cap B \cap C)}{P(B \cap C)}$$

A regra fundamental

$$P(A|B)P(B) = P(A \cap B)$$
$$P(A|B \cap C) = P(A \cap B|C)$$

Regra de Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$
$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}$$

$P(A,B) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(A = a_i, B = b_j) = 1.$

A regra fundamental para variáveis

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{P(A,B)}{\sum_B P(A,B)}$$
$$P(B|A,C) = \frac{P(A|B,C)P(B|C)}{P(A|C)} = \frac{P(A,B|C)}{\sum_B P(A,B|C)}$$

Se $P(A|C,B) = P(A,B)$ *então* A *e* C *são condicionalmente independentes dado* B .

$P(A,B,C) = P(A|B,C)P(B,C)$

O Capítulo α4 foca no modelo de Agente BDI Bayesiano.

α4 AGENTES BDI BAYESIANOS

α4.1 Introdução

A Inteligência Artificial frequentemente depara-se com situações em que facilmente torna-se impraticável lidar com todas as possibilidades. Apesar de haver cenários para os quais incerteza não precisa ser modelada, estes podem ser considerados um subconjunto dos cenários do mundo real. O próprio fato de que incerteza permeia a realidade serve como evidência da aplicabilidade limitada de um modelo de agente autônomo que não lida com o desconhecido. A fim de lidar com esta demanda, Fagundes e Vicari (FAGUNDES; VICARI, 2007) realizam um esforço para integrar o modelo de agente BDI e Redes Bayesianas. Adicionalmente, Kieling e Vicari (KIELING; VICARI, 2010) integram Redes Bayesianas a uma implementação do interpretador AgentSpeak(L) (RAO; GEORGEFF, 1995) *Jason* (BORDINI; HÜBNER; WOOLDRIDGE, 2007); nesse trabalho, desejos não recebem uma representação explícita.[1] Finalmente, Carrera e Iglesias (CARRERA; IGLESIAS, 2012) focam no processo de atualização de crenças dentro de um agente BDI bayesiano.

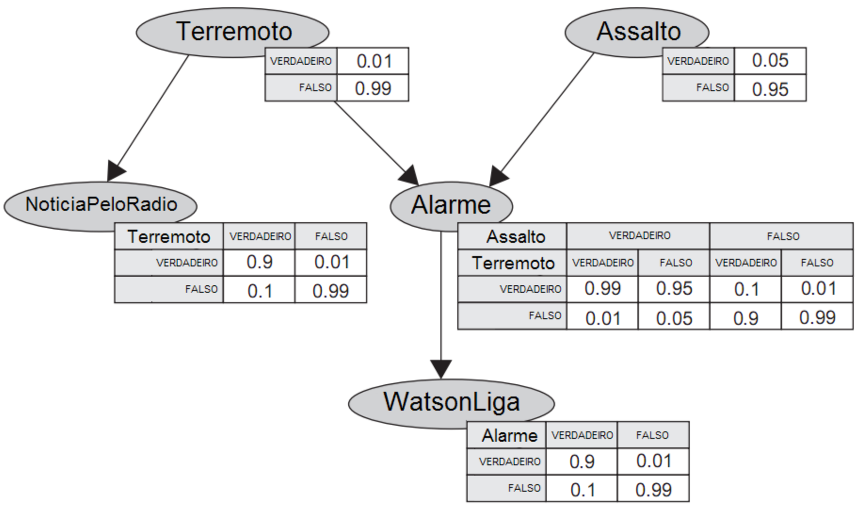
Este capítulo tem por foco o Objetivo 4 (os objetivos estão listados na Seção α1.1), i.e., apresentar um modelo de agente BDI bayesiano.

α4.2 Exemplo ilustrativo

Consideremos o problema do *Assalto ou Terremoto* (*"Burglary or Earthquake"*) (PEARL, 1988), o qual usamos posteriormente. O Sr. Holmes está trabalhando em seu escritório e então recebe uma ligação do Sr. Watson, que conta a ele que seu alarme disparou. Acreditando que assaltaram sua casa, o Sr. Holmes corre para casa. No caminho para casa, ele liga o rádio e ouve notícias sobre um pequeno terremoto que aconteceu na área. Ciente de que terremotos tendem a disparar alarmes de invasão, ele volta ao trabalho. A Figura α4.1 (FAGUNDES; VICARI, 2007) é um exemplo de Rede Bayesiana que ilustra este problema.

Nessa Rede Bayesiana, é dito que a probabilidade de um $Terremoto$ ocorrer é de 1%, e que a probabilidade de um $Assalto$ é de 5%. Se for observado que um $Terremoto$ ocorreu, é dito haver uma probabilidade de 90% do Sr. Holmes ouvir $NoticiaPeloRadio$ a respeito, e uma probabilidade de 1% dele ouvir esta notícia sem ter havido um $Terremoto$ (i.e., notícias falsas). Adicionalmente, se um $Terremoto$ ocorre e um $Assalto$ de fato acontece, a probabilidade do $Alarme$ na casa do Sr. Holmes disparar é de 99%, enquanto se é sabido que apenas o $Terremoto$ ocorreu, a probabilidade do $Alarme$ ser ativado é de

[1]Como também é o caso com o próprio Jason (MENEGUZZI, 2009).



Terremoto	
VERDADEIRO	0.01
FALSO	0.99

Assalto	
VERDADEIRO	0.05
FALSO	0.95

Terremoto	VERDADEIRO	FALSO
VERDADEIRO	0.9	0.01
FALSO	0.1	0.99

Assalto	VERDADEIRO		FALSO	
Terremoto	VERDADEIRO	FALSO	VERDADEIRO	FALSO
VERDADEIRO	0.99	0.95	0.1	0.01
FALSO	0.01	0.05	0.9	0.99

Alarme	VERDADEIRO	FALSO
VERDADEIRO	0.9	0.01
FALSO	0.1	0.99

Figura α4.1: Rede Bayesiana para *Assalto ou Terremoto*

apenas 1%. Um $Assalto$ acontecendo sem o $Terremoto$ resulta numa probabilidade de 95% do $Alarme$ disparar. Se nem um $Terremoto$, nem um $Assalto$ acontece, é afirmado que a probabilidade do $Alarme$ disparar é de 1%.

α4.3 Tipos de conexão e dependências entre variáveis

A Figura α4.2 (FAGUNDES; VICARI, 2007) apresenta instâncias dos três tipos de conexão presentes na Rede Bayesiana na Figura α4.1[2]. Se o estado da variável $Alarme$ é conhecido (i.e., há *evidência* dele), então evidência sobre $Assalto$ não afetará $WatsonLiga$ (i.e., $Assalto$ e $WatsonLiga$ são *d-separados*). De outro modo, crenças sobre $Assalto$ influenciam crenças sobre $WatsonLiga$. Se é observado que o $Alarme$ disparou, as crenças sobre $WatsonLiga$ são atualizadas. Da mesma forma, pode-se atualizar as crenças sobre o $Alarme$ se o estado de $WatsonLiga$ é observado (raciocínio abdutivo). Além disso, se sabe-se que o $Alarme$ disparou, isto significa que as chances de ter havido um $Terremoto$ são afetadas (*d-conectados* através de uma conexão serial), assim como as de ouvir $NoticiaPeloRadio$ a respeito ($Alarme$ e $NoticiaPeloRadio$ são *d-conectados* através de uma conexão divergente vinda do (não-instanciado) $Terremoto$). Isto também significa que $Terremoto$ e $Assalto$ afetam um ao outro; se um ocorreu, as chances do outro também ter ocorrido são reduzidas (*d-conectados* através de uma conexão convergente para o (instanciado) $Alarme$). Se o estado de $Alarme$ não é conhecido, $Terremoto$ e $Assalto$ não causam impacto um ao outro.

α4.4 Modus operandi

Delineamos um ciclo de raciocínio genérico para um agente BDI bayesiano situado em um ambiente incerto, no Algoritmo 2. Primeiramente, o agente atualiza sua base de crenças, i.e., sua Rede Bayesiana, de acordo com o que ele percebeu do ambiente (Linha 2). Em segundo lugar, o agente avalia se cada desejo foi satisfeito, removendo-o da lista de desejos caso sim (Linha 3). O agente então prossegue para avaliar seus possí-

[2]A Seção α3.3 cobre os três tipos de conexão encontrados em Redes Bayesianas e o critério de *d-separação* que indica se evidência pode ser transmitida entre variáveis.

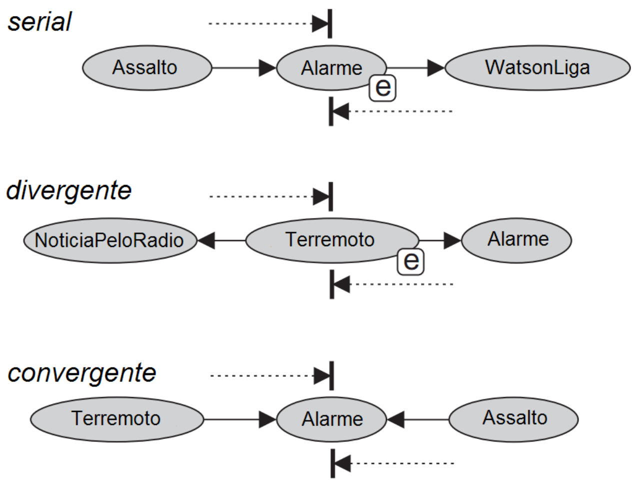


Figura α4.2: *Assalto ou Terremoto* – tipos de conexão

Algoritmo 2 Ciclo de Raciocínio para agentes BDI bayesianos

```
 1: procedure CICLO DE AGENTE BDI BAYESIANO
 2:     atualizar crenças baseando-se em percepções
 3:     avaliar satisfação dos desejos, remover desejos satisfeitos
 4:     avaliar e possivelmente escolher desejos
 5:     buscar planos que possam satisfazer os desejos escolhidos
 6:     para cada desejo escolhido, if um plano aplicável foi encontrado, criar uma in-
    tenção e associá-la com o desejo e com o plano, que é consequentemente adotado e
    executado
 7:     se um plano não foi encontrado, marcar o desejo como insatisfatível no momento
 8:     se plano adotado falhou, procurar outro ou remover a intenção (sujeito à política
    de compromisso)
 9:     se plano adotado teve sucesso, remover intenção
10: end procedure
```

veis desejos (Linha 4) – o processo de seleção de desejos propriamente dito é assunto do Capítulo α5). Se um desejo foi selecionado, o agente deve tentar adotar a intenção de satisfazê-lo, i.e., comprometer-se com o desejo. Com isto em mente, ele busca planos que aparentem ser capazes de satisfazê-lo através de uma sequência de ações (Linha 5). Nós dizemos "aparentem" porque um plano pode acabar sendo ineficaz, apesar de uma validação pré-condicional bem sucedida, já que as pré-condições propriamente ditas podem ter sido incertas (elas foram, a não ser que tenha havido evidência forte acerca delas), *e* pode ter havido mudanças inesperadas no ambiente (e.g., causadas por outros agentes). Tendo obtido um conjunto de planos aplicáveis, para cada desejo escolhido para o qual um plano aplicável foi obtido, uma intenção é criada e associada a ele. Esta intenção é então associada também ao plano que espera-se satisfazê-la, com o agente efetivamente "adotando" tal plano, o qual é executado por ele (Linha 6). Para cada desejo escolhido para o qual nenhum meio de satisfazê-lo foi encontrado, marcá-lo como "insatisfatível neste momento" e abster-se de criar uma intenção para ele no ciclo corrente (Linha 7). Se há um plano adotado e este falha, o agente pode buscar um plano alternativo ou abandonar a intenção de uma vez (Linha 8). Isto está sujeito a uma *política de comprometimento*. Se uma execução de plano adotado é completada com sucesso, a intenção correspondente é removida (Linha 9).

α4.5 Crenças

Nós consideramos que as crenças correspondem à Rede Bayesiana inteira, assim como fazem Kieling e Vicari (KIELING; VICARI, 2010). Nós dizemos que há uma crença de que há uma probabilidade p que uma dada variável x esteja no estado s. Cada variável de evento tem n possíveis estados, cada um com uma probabilidade associada, que ou está prontamente disponível em uma *Tabela de Probabilidades Condicionais* se o estado de todas as variáveis condicionantes é conhecido (i.e., há evidência sobre cada uma), ou deve ser obtido (i.e., calculado) via inferência bayesiana.

α4.6 Desejos

Desejos no modelo de agente BDI bayesiano correspondem a estados específicos de variáveis de eventos na Rede Bayesiana. Cada desejo possui uma crença pré-condicionante, seguindo a tradição de muitos sistemas BDI implementados (e.g., (RAO, 1996), (MÓRA et al., 1999)).

Enquanto considerando o conceito de desejos junto a Redes Bayesianas, contemplamos duas abordagens divergentes. Já que no modelo BDI original desejos podem ser considerados o que o agente gostaria que fosse verdade, o que em contrapartida refletiria-se em suas crenças, cogitamos estender esta noção para agentes BDI bayesianos considerando “aquilo que é verdadeiro” intersubstituível com “aquilo que é suportado por evidência”. Isto essencialmente significaria que apesar das crenças poderem ser incertas, o status de cada desejo não.

Todavia, já que isso configura uma restrição considerável, cremos que não deveria ser abraçada sem resistência. O conceito de desejo (um estado desejado para uma variável) que poderia ser considerado satisfeito sem *real confirmação* (i.e., evidência forte), possivelmente por meio de uma avaliação de probabilidade baseada em limiar (e.g., uma probabilidade de 80% que você não tenha uma certa doença baseando-se em resultados de exames, não impossibilita um falso negativo, mas também não é um valor fácil de se desconsiderar) poderia viabilizar agentes mais flexíveis, *mas* tais agentes às vezes estariam “fora de contato com a realidade”, acreditando e considerando ter atingido supostos fatos desejados que simplesmente não seriam verdade.

Em um ser humano de verdade, esta característica poderia até mesmo fazer o indivíduo ser considerado delirante. Por outro lado, esta relação "com ruído"com o ambiente potencialmente permite maior eficiência, já que cobre suposições ao invés de depender de confirmações – que poderiam ser desde totalmente inviáveis até extremamente custosas. Assim, um contra-argumento ao rótulo de delírio poderia ser que um certo nível dessa atitude permissiva é essencial para não haver uma condição que poderia ser descrita como “transtorno obsessivo-compulsivo”, já que requerer confirmação para cada desejo pode ser incompatível com cenários realistas. Dados os dois pontos de vista, acreditamos que a calibragem do nível de tolerância à incerteza interna no processo de avaliação de satisfação de desejos é o fator que essencialmente regula comportamento são (i.e., funcional).

Propomos duas categorias de desejos diferindo na avaliação de satisfação: desejos *fortes* e desejos *fracos*.

α4.6.1 Desejos fortes

Desejos fortes são desejos para os quais deve haver evidência forte para que eles possam ser considerados satisfeitos. Não pode haver qualquer dúvida, não importa o quão pequena, quanto ao desejo forte ter sido satisfeito. Este é um caso especial do que de outro modo seria um desejo fraco (descrito na Seção α4.6.2), onde, dado um desejo (i.e., um estado de variável de evento) d, $P(d) = 1$.

Logo, enquanto neste modelo de agente há uma preocupação com suportar incerteza, para este tipo de desejo a incerteza não é aceitável no processo de raciocínio do agente.

α4.6.2 Desejos fracos

Desejos fracos são aqueles que não se deve esperar que necessariamente sejam confirmados, i.e., por evidência forte, mas que pelo menos seja suficientemente provável que eles tenham sido satisfeitos. Desejos fracos podem ser vistos como desejos que aceitam *evidência fraca* como suficiente para serem considerados satisfeitos. O processo de avaliação de satisfação para desejos fracos é baseado em limiar.

α4.7 Intenções

Similarmente a agentes BDI tradicionais, intenções são desejos com os quais o agente se comprometeu. O agente buscará um plano aplicável à situação corrente: qualquer plano voltado a satisfazer pelo menos uma das intenções e cujas pré-condições não forem conclusivamente negadas (i.e., pré-condições não forem contraditórias à evidência forte) é válido.

Nós acreditamos que é indispensável ser capaz de determinar se um plano de ações falhou ou não. O estado de pré-condições de planos *pode*, entretanto, ser incerto, conceitualmente aumentando a suscetibilidade à falha durante a execução. Em outras palavras, já que incerteza é um fator constante a ser levado em conta no modelo corrente, falha em planos resultante dela deveria ser vista como *iminente*.

Idealmente, um algoritmo gerador de planos que funcionasse com este modelo de agente estaria disponível, junto com uma biblioteca de planos. Este é um assunto fora do escopo deste trabalho, contudo, e aqui a solução mais prática é supôr que todos os planos sejam pré-definidos.

É válido observar que as pré-condições terem sido confirmadas (i.e., suportadas por evidência forte) ou não pode ser relevante no caso de falha na execução de um plano, já que tal falha pode resultar do fato de que o plano na verdade nunca deveria ter sido usado (embora não estivéssemos cientes disto no momento de sua adoção) ou não.

α4.8 Exemplo

A Rede Bayesiana de *Assalto ou Terremoto* (Figura α4.1) pode ajudar a ilustrar um exemplo. A rede em questão corresponde às crenças do agente AoT. Se Watson (e.g., outro agente) chamar AoT, é bastante provável que o alarme tenha sido disparado – independentemente da causa – como pode ser visto na *Tabela de Probabilidades Condicionais* para a variável $WatsonLiga$.

Definimos o desejo forte $D1(Alarme = falso)$, condicionado a $WatsonLiga = verdadeiro$, i.e., um desejo de ter o $Alarme$ desligado (i.e., crer que o alarme esteja desligado), o que neste exemplo nós presumimos implicar no plano de avaliar a situação em casa, mediante a ligação de Watson. A razão para esta condição é que nosso agente crê ser altamente provável (94.2%) que o alarme já esteja desligado em circunstâncias normais – inferência bayesiana [3] com um conjunto de evidências vazio reflete isto:

$$P(Alarme) = \sum_{Terremoto,Assalto} P(Alarme, Terremoto, Assalto)$$

$$=\sum\nolimits_{Terremoto,Assalto} P(Alarme|Terremoto,Assalto)P(Terremoto,Assalto)$$

$$=P(Alarme|Terremoto{=}verdadeiro,Assalto{=}verdadeiro)P(Terremoto{=}verdadeiro,Assalto{=}verdadeiro)+$$

$$P(Alarme|Terremoto{=}verdadeiro,Burlgary{=}falso)P(Terremoto{=}verdadeiro,Assalto{=}falso)+$$

$$P(Alarme|Terremoto{=}falso,Assalto{=}verdadeiro)P(Terremoto{=}falso,Assalto{=}verdadeiro)+$$

$$P(Alarme|Terremoto{=}falso,Assalto{=}falso)P(Terremoto{=}falso,Assalto{=}falso)$$

$$=(0.99,0.01)\cdot(0.01\cdot0.05)+(0.1,0.9)\cdot(0.01\cdot0.95)+(0.95,0.05)\cdot(0.99\cdot0.05)+(0.01,0.99)\cdot(0.99\cdot0.95)$$

$$=(0.99,0.01)\cdot0.0005+(0.1,0.9)\cdot0.0095+(0.95,0.05)\cdot0.0495+(0.01,0.99)\cdot0.9405$$

$$=(0.000495,0.000005)+(0.00095,0.00855)+(0.047025,0.002475)+(0.009405,0.931095)$$

$$=(0.057875,\mathbf{0.942125})$$

Observe que $P(Terremoto, Assalto) = P(Terremoto) \cdot P(Assalto)$ porque $Terremoto$ e $Assalto$ são ***d-separados*** em sua conexão ***convergente*** para $Alarme$, i.e., eles são condicionalmente independentes dada nenhuma evidência forte em qualquer um de seus descendentes.

Nós também definimos o desejo forte $D2(Assalto = falso)$, condicionado a $Alarme = verdadeiro$. Isto significa, como uma simples questão de decisão de projeto do agente, que nosso agente somente se importa em determinar que não houve $Assalto$ na medida em que acredita que o $Alarme$ foi ativado.

Nosso agente possui um plano de ações $P1$, que envolve ir para casa e determinar com certeza o estado do alarme, assim como se alguém assaltou a casa, i.e., obter evidência sobre o estado atual das variáveis $Alarme$ e $Assalto$. Isto ilustra como um plano de ações tem como propósito a obtenção de evidência, para que um desejo possa ser satisfeito ou não. O plano $P1$ pode servir para ambos os desejos, caso estes sejam selecionados, permitindo que intenções correspondentes possam ser, consequentemente, criadas.

A seleção de $D1$ e $D2$ não deve depender de evidência forte de $Alarme = verdadeiro$, pois esta pode simplesmente não ficar disponível. Em outras palavras, seleção de desejos para agentes BDI tradicionais, dependente de simples avaliações lógicas, não nos ajuda aqui.

No Capítulo $\alpha5$ cobrimos Seleção de Desejos BDI Bayesianos.

[3] A ferramenta interativa JavaBayes (COZMAN, 2000) pode ser usada para facilitar cálculos como este.

α5 SELEÇÃO DE DESEJOS BDI BAYESIANOS

α5.1 Introdução

Conforme mencionado no Capítulo α2, agentes BDI tradicionais selecionam desejos para adoção como intenções baseados no estado de sua base de crenças, e as restrições que indicam se um dado desejo é viável e passível de ser adotado como uma intenção podem ser expressas como expressões lógicas sobre literais *ground*. Esta abordagem, entretanto, não é capaz de lidar com informação incerta.

Em agentes BDI bayesianos, crenças são expressas como estados de variáveis de eventos em Redes Bayesianas ao invés de como literais *ground*, levando à necessidade de mecanismos mais elaborados para a seleção de desejos. Fagundes et al. (FAGUNDES; VICARI; COELHO, 2007) propõem um tal mecanismo. Nesta dissertação, apresentamos uma limitação que tal mecanismo possui, e provemos mecanismos de seleção de desejos BDI que se encaixam em agentes BDI bayesianos e não possuem tal limitação, i.e., o Objetivo 5 (os objetivos estão listados na Seção α1.1).

α5.2 Crenças e Desejos

Em agentes BDI tradicionais, um desejo é comumente representado como uma regra lógica (e.g., $crenca \Rightarrow desejo\_valido$). Nós o representamos como a seguir:

<desejo> : { <lista de crenças> } ,

onde *desejo* denota o nome do desejo e *lista de crenças* é uma lista de crenças, separadas por vírgula, que corresponde aos requisitos para este desejo não ser considerado a priori conceitualmente inadequado para seleção. Cada crença é geralmente um literal *ground*, o qual é então avaliado para *verdadeiro* ou *falso*.

As crenças de agentes BDI bayesianos são estendidas com dados probabilísticos – cada crença corresponde a um estado de variável de evento numa Rede Bayesiana com o qual uma probabilidade é associada. Assim, não é mais suficiente realizar a avaliação lógica anterior de crenças requeridas para cada desejo potencialmente elegível para criação de intenção; em outras palavras, para agentes BDI bayesianos, um procedimento criado para agentes BDI tradicionais para determinar se a condição que indica se um desejo deve ser selecionado para ter uma intenção associada foi satisfeita é limitado num contexto probabilístico, já que ele não diferencia entre graus de probabilidade associados com pré-condições de desejos. Adicionalmente, assim como há graus de probabilidade para as crenças, selecionar um desejo é agora uma decisão feita com graus variáveis de confiança, o que implica que pré-condições não mais se tratam estritamente de *validade* de seleção, mas também de *confiança* numa seleção que é inerentemente incerta. O único caso em que pré-condições de desejos ainda constituem uma questão de validade é quando há evidência forte *contra* a pré-condição em questão (i.e., evidência de um estado diferente da variável de evento).

α5.3 Seleção de Desejos

Como previamente afirmado, o processo de seleção de desejos em agentes BDI bayesianos requer tratamento especial. A não ser que haja evidência do estado das crenças pré-condicionantes, não sabemos se selecionar um desejo é realmente apropriado. Não obstante, devemos proceder em meio à incerteza.

α5.3.1 Seleção Baseada em Limiar

A fim de permitir que um agente gerencie seleção de desejos com incerteza, Fagundes et al. (FAGUNDES; VICARI; COELHO, 2007) propõem selecionar desejos seguindo um critério baseado em limiar , onde um desejo é elegível para seleção se a probabilidade pré-condicionante for igual ou maior que um determinado valor de limiar. Entretanto, utilizar um limiar probabilístico como critério efetivamente desconsidera quaisquer desejos que simplesmente não sejam suficientemente promissores quanto ao quão provável é que seus requisitos de fato sejam válidos no mundo.

No que se refere à seleção de desejos condicionados a crenças incertas, acreditamos que seleção baseada num dado limiar de probabilidade é insuficiente, porque pode haver um subconjunto de desejos que nunca satisfarão tal requisito. Consequentemente, investigamos e apresentamos abordagens adicionais para a seleção de desejos em agentes BDI bayesianos a partir da próxima seção.

α5.3.2 Ranking de Probabilidades

Nossa primeira abordagem envolve ordenar a lista de desejos por probabilidade, em ordem decrescente; deste modo temos um ranking indo do desejo com a pré-condição com a maior probabilidade ao desejo com pré-condição com a menor probabilidade, para o processo de seleção de desejos. Agora pegamos o primeiro desejo da lista (i.e., o desejo mais provável de ser viável), e criamos uma intenção para ele. Esta intenção é associada a um plano, obtido ou de uma biblioteca de planos ou sob demanda usando um algoritmo de planejamento.

Visto que um agente BDI bayesiano é inerentemente esperado estar situado em um ambiente com incerteza, a possibilidade de suas intenções falharem, incluindo, mas não exclusivamente devido a suposições incorretas acerca de crenças *deve* ser levada em consideração. O que acontece neste caso é sujeito à política adotada pelo agente – seguem exemplos de como um agente BDI bayesiano pode se comportar:

- Um agente pode ser "teimoso" e manter o desejo selecionado, esperando que a intenção não falhe na próxima vez. Neste caso, é sábio impôr um limite de quantas vezes um desejo associado a uma intenção que acabou de falhar pode ser mantida em detrimento do próximo desejo na lista, a fim de que não deliberadamente incorramos no problema do "desejo que nunca é escolhido". Observe que insistir em um desejo que falha em ser satisfeito devido a suposições acerca de suas pré-condições que na verdade são falsas (i.e., seleção que não seria feita se não fosse pela incerteza das crenças pré-condicionantes) deve continuar resultando em falhas;

- O desejo que falhou em ser satisfeito pode cair algumas posições no ranking de acordo com uma fórmula pré-definida, permitindo que o próximo desejo na lista seja escolhido;

- Cada um dos desejos pode ter um contador de falhas ou ser armazenado em listas, cada uma associada a um número de falhas, e tal critério pode ser usado para decidir o próximo desejo a ser selecionado;

- O desejo que não pôde ser satisfeito pode simplesmente ser reinserido no final da lista, o agente deste modo sendo justo para com o desejo até então último colocado no ranking.

Observe que apesar do fato da lista de desejos ser originalmente criada como um ranking probabilístico, estas estratégias fazem com que a ordem não tenha que realmente refletir o conceito em todos os momentos (e.g., o desejo pré-condicionado por uma crença com a maior probabilidade poderia estar bem no final da lista em um determinado momento como o resultado de uma falha de plano que levou a sua intenção associada a ser descartada).

Se o plano falhar, da mesma forma poderá a intenção, ou um novo plano pode ser buscado. Se ele tiver sucesso, o mesmo será dito da intenção e da satisfação do desejo associado. Especialmente se não há intenção corrente, devemos executar o procedimento de seleção de desejos novamente em breve. Graças a este ranking, mesmo o desejo com a menor probabilidade de ser viável será selecionado em algum momento, levando uma intenção voltada a satisfazê-lo (i.e., nenhum desejo é absolutamente irrelevante).

Um problema com esta abordagem é que o custo de perseguir uma intenção (i.e., executar um plano associado a uma intenção) pode ser considerável, e desejos bastante improváveis de serem satisfeitos são escolhidos facilmente, ainda que obedecendo uma ordenação por prioridade. Isto é algo que percebemos enquanto considerando possíveis projetos de agente. Visando oferecer uma solução que pareça mais sensata e ainda não prejudicada como a solução previamente proposta do limiar, exploramos ainda outra possibilidade.

α5.3.3 Loteria Viciada

Esta abordagem implica inserir a noção de probabilidade no processo de seleção de desejos propriamente dito. A ideia é aleatoriamente gerar um número e usá-lo para determinar qual desejo escolher, de acordo com uma distribuição de probabilidades que apropriadamente reflita as probabilidades das pré-condições de cada desejo. Geramos intervalos numéricos de tamanhos condizentes com as probabilidades, em $[0, 1]$. Se a soma das probabilidades em questão for maior que 1, normalizamo-as. As probabilidades servem como pesos que criam viés no que de outra maneira constituiria uma seleção puramente aleatória. É uma seleção de desejos não-determinística que é sujeita a viés da distribuição de probabilidades.

Este método de seleção de desejos nem desconsidera desejos suportados por crenças dotadas de probabilidades muito baixas, nem potencialmente abraça-as mais frequentemente do que o senso comum permitiria – do que tais probabilidades sugeririam.

Cada valor decimal de ponto flutuante aleatoriamente gerado no intervalo $[0, 1]$ resultará em um dos desejos ser escolhido, respeitando os graus de certeza de suas respectivas crenças pré-condicionantes. Se um desejo não pode ser satisfeito em um dado momento, simplesmente realiza-se a seleção de desejos através da geração de um valor aleatório novamente.

Nós não realizamos normalização quando a soma das probabilidades das pré-condições é menor que 1.0, já que isto inflaria as probabilidades de seleção para desejos pré-condicionados a eventos insignificantes. Por exemplo, havendo um único desejo, pré-condicionado a uma crença com uma probabilidade de 0.0001, ele seria tratado como se tal probabilidade fosse 1.0. Observe que como os intervalos numéricos não interseccionam uns com os outros, já que o valor numérico aleatório deve pertencer a, no máximo, um tal intervalo. Isto possui relação com o fato de que espera-se que não mais que um desejo seja selecionado.

α5.3.4 Seleção Aleatória Multidesejos com Viés

Ao trabalhar com Loteria Viciada, consideramos ainda outra abordagem para a seleção de desejos. Até agora, trabalhamos sob a suposição de que cada ciclo de seleção resultaria em no máximo um desejo. Agora consideramos que pode ser uma boa ideia *selecionar múltiplos desejos de uma só vez.*

Esta abordadem para seleção de desejos remove competição entre desejos, contanto que eles não conflitem, considerando tais desejos não-conflitantes independentemente uns dos outros. Consideramos que existe um conflito entre dois desejos se eles referem-se à mesma variável de evento, mas a diferentes estados – todos estados no espaço de estados de uma variável de evento são mutuamente exclusivos. Se um dado desejo sendo avaliado conflita com um desejo que já foi designado a ser selecionado no final do processo de seleção corrente, consideramos-no inelegível para seleção. Caso contrário, ele recebe uma chance: dado um desejo D_i pré-condicionado a uma crença associada a uma probabilidade P_i, dizemos que é atribuído a D_i um intervalo numérico $I_i = [0, P_i]$. Para cada tal desejo D_i, se um valor numérico gerado aleatoriamente N_i no intervalo $[0, 1]$ pertence ao intervalo I_i, o desejo é adicionado ao conjunto de desejos a serem selecionados no final deste ciclo de seleção. Opcionalmente, podemos ordenar a lista em ordem crescente de probabilidade da pré-condição, antes da execução do algoritmo, para permitir que um desejo suportado por uma probabilidade menor possa receber sua chance sem ser prejudicado por desejos potencialmente conflitantes que tenham uma chance maior de ser selecionados. Se a diferença de probabilidades de pré-condição de desejos conflitantes não for grande, todavia, isto pode ser considerado um viés indesejável, já que fará com que um desejo de seleção supostamente mais improvável seja selecionado mais frequentemente do que desejo(s) conflitante(s) na prática. Outra opção seria simplesmente embaralhar os desejos.

A vantagem que enxergamos nesta abordagem, se comparada à Loteria Viciada, é que não há chances reduzidas de seleção para cada desejo em casos onde haveria normalização, i.e., em situações onde a lista de desejos permitiria uma soma de probabilidades de pré-condições maior que 1, o que por sua vez levaria a probabilidades individuais de seleção de desejos reduzidas.

Seleção Aleatória Multidesejos com Viés garante que, no contexto do processo de seleção de desejos propriamente dito, desejos não-conflitantes não podem se influenciar mutuamente no que diz respeito a sua probabilidade de serem selecionados, já que são tratados independentemente uns dos outros (e.g., paralelismo completo é possível). Por outro lado, adotar múltiplos desejos de uma vez só *poderia* ser oneroso; dependeria de requisitos de planos específicos para cada cenário particular. De qualquer modo, cremos que a abordagem é relevante no escopo deste trabalho.

α5.4 Exemplo Concreto: Agente Vigia

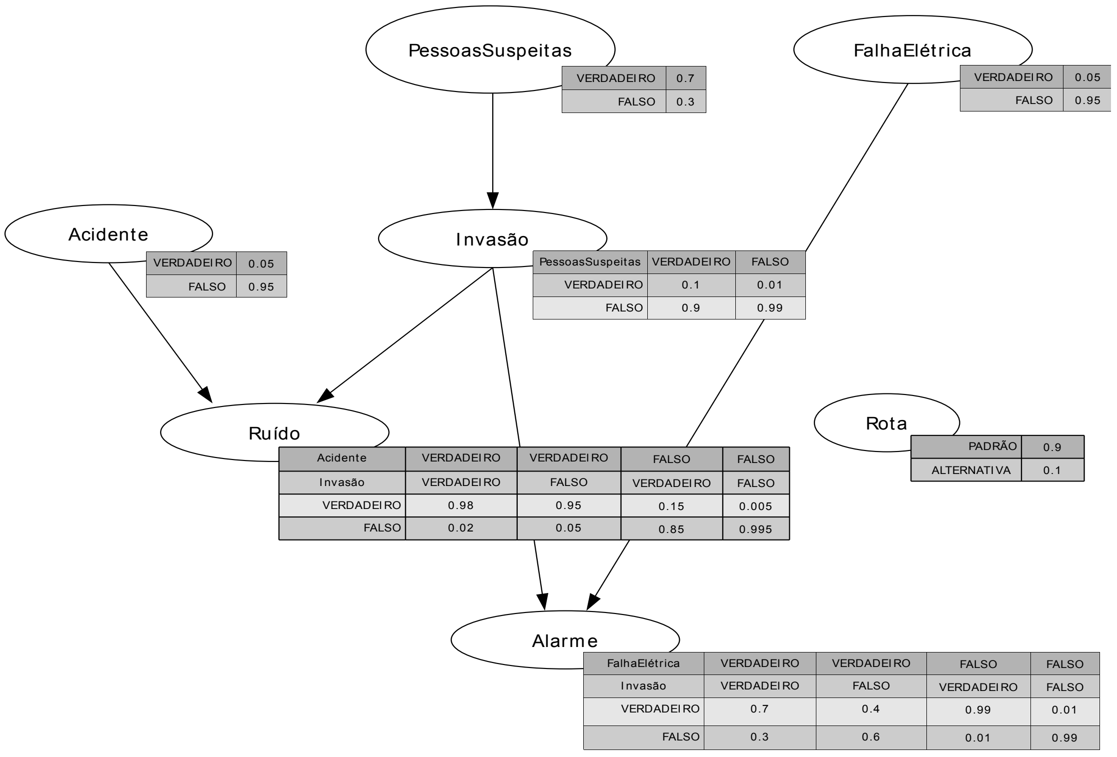
A fim de ilustrar os efeitos de cada mecanismo de seleção de desejos descrito na Seção α5.3, agora apresentamos um exemplo concreto para mostrar como um agente pode reagir a situações usando os algoritmos apresentados. Nosso cenário de exemplo consiste de um agente encarregado de vigiar uma instalação e reportar qualquer coisa fora do comum. A presença de pessoas suspeitas nas redondezas aumenta sua estimativa de uma invasão. Há um alarme na instalação, o qual é eficaz em circunstâncias normais. Entretanto, há relatos de falhas elétricas ocasionais na instalação, as quais podem fazer com que o alarme dispare por nenhuma razão ou não dispare quando deveria. Além disso, o vigia fica interessado em obter evidência de que não há falha elétrica se ele sabe que há pessoas suspeitas nas proximidades. Se uma falha elétrica é detectada, o vigia deve reportar isto. A área em torno da instalação é conhecida pelo tráfego intenso, e acidentes são mais comuns que o normal, resultando em ruído que quase sempre é percebido pelo agente. Todavia, ruído pode ser causado por invasores, apesar disto não ser muito provável. Se o vigia descobre que um acidente ocorreu, isto deve ser reportado também (e.g., para que outros tenham isso em mente se ouvirem ruído). Visando patrulhar a instalação, o vigia periodicamente escolhe entre a rota padrão e uma alternativa, e fica mais inclinado a patrulhar a rota alternativa conforme sua crença de que uma invasão ou é iminente ou está acontecendo aumenta, e, pelo contrário, o vigia fica mais inclinado a patrulhar a rota padrão quando tudo parece calmo. O vigia também deve ficar atento a falhas elétricas e acidentes.

No que se refere às relações entre as variáveis de eventos na rede, observamos que: *i)* Evidência da presença de pessoas suspeitas nas proximidades aumenta a probabilidade de uma invasão; *ii)* Evidência de ativação do alarme aumenta a probabilidade de ocorrência de uma invasão, assim como a probabilidade de haver pessoas suspeitas nas proximidades. Isto ainda é verdade se também há evidência de uma falha elétrica, mas o aumento de probabilidade para ambos estados de variáveis de eventos é menor. Se há evidência de que não há falha elétrica (e.g., uma notificação sobre manutenção recentemente realizada), o aumento de probabilidade é o maior dos três casos; *iii)* Evidência de ruído aumenta a probabilidade de uma invasão. Contudo, este aumento é quase anulado mediante evidência de um acidente, já que esta rede nos diz que um acidente é uma causa muito mais provável de ruído que uma invasão, e o impacto de uma invasão na probabilidade de ruído se não já sabemos que houve um acidente é pequeno; e *iv)* Um aumento na probabilidade de uma invasão (e.g., através de evidência de pessoas suspeitas e ruído) aumenta a probabilidade de ativação do alarme, mesmo se há uma falha elétrica, apesar de que neste caso o aumento de probabilidade é menor.

A Rede Bayesiana que codifica o conhecimento do domínio descrito no cenário está representada na Figura α5.1, e ela representa as crenças do agente *Vigia*. Observe que não associamos a variável $Rota$ a uma crença sobre o estado do ambiente, mas sim a uma crença interna associada à rota atualmente adotada pelo agente. Ela não é uma parte do raciocinio em torno da probabilidade de uma invasão or qualquer uma das outras variáveis de eventos, e esta é a razão pela qual deixamos-na desconectada de todos os outros nodos na rede (i.e., a rota não é modelada como uma causa ou consequência lógica de qualquer evento nela).

Este agente vigia possui dois desejos fortes mutualmente exclusivos que são periodicamente renovados:[1]

[1]Denotamos desejos na forma `<desejo>(<crença pré-condicionante>)`, onde ambos elementos



PessoasSuspeitas	
VERDADEIRO	0.7
FALSO	0.3

FalhaElétrica	
VERDADEIRO	0.05
FALSO	0.95

Acidente	
VERDADEIRO	0.05
FALSO	0.95

PessoasSuspeitas	VERDADEIRO	FALSO
VERDADEIRO	0.1	0.01
FALSO	0.9	0.99

Rota	
PADRÃO	0.9
ALTERNATIVA	0.1

Acidente	VERDADEIRO	VERDADEIRO	FALSO	FALSO
Invasão	VERDADEIRO	FALSO	VERDADEIRO	FALSO
VERDADEIRO	0.98	0.95	0.15	0.005
FALSO	0.02	0.05	0.85	0.995

FalhaElétrica	VERDADEIRO	VERDADEIRO	FALSO	FALSO
Invasão	VERDADEIRO	FALSO	VERDADEIRO	FALSO
VERDADEIRO	0.7	0.4	0.99	0.01
FALSO	0.3	0.6	0.01	0.99

Figura α5.1: Rede Bayesiana correspondente às crenças do agente *Vigia*

- $Rota.padrao(Invasao.falso)$; e
- $Rota.alternativa(Invasao.verdadeiro)$.

Isto é, o agente deseja patrulhar a rota padrão se não há uma invasão, e a rota alternativa caso contrário. Ele também possui os desejos fortes $FalhaEletrica.falso($ $PessoasSuspeitas.verdadeiro)$ – o desejo de crer que não há falha elétrica no momento, condicionado à probabilidade de haver pessoas suspeitas nas proximidades – e $Acidente.verdadeiro(Ruido.verdadeiro)$ – o desejo de descobrir que houve um acidente, se ruído foi ouvido.

Agora apresentamos brevemente o resultado do uso de cada um dos quatro algoritmos enquanto trabalhando com um cenário inicial em que não há evidência forte de nenhum evento. Já que não há evidência neste momento, $P(PessoasSuspeitas) = (0.7, 0.3)$ (i.e., $P(PessoasSuspeitas = verdadeiro) = 0.7$ and $P(PessoasSuspeitas = falso) = 0.3$), e consequentemente $P(Invasao) = (0.073, 0.927)$; adicionalmente, $P(Ruido) = (0.06241525, 0.93758475)$. O desejo $Rota.padrao$ é pré-condicionado à crença que $Invasao$ é $falso$, a qual possui uma probabilidade de 0.927; o desejo $Rota.alternativa$ é pré-condicionado à crença que $Invasao$ é $verdadeiro$, a qual possui uma probabilidade de 0.073; o desejo $FalhaEletrica.falso$ é pré-condicionado à crença que $PessoasSuspeitas$ é $verdadeiro$, a qual possui uma probabilidade de 0.7; e o desejo $Acidente.verdadeiro$ é pré-condicionado à crença que $Ruido$ é $verdadeiro$, a qual possui uma probabilidade de 0.06241525.

Primeiramente, consideremos seleção baseada em limiar, com um limiar de 0.75. Isto significa $Rota.padrao(Invasao.falso)$ é um desejo elegível para seleção, já que $P(Invasao = falso) = 0.927$. Em um cenário onde haja evidência forte de ruído

são descritos na forma <`variável de evento`>.<`estado`>.

(i.e., $P(Ruido = verdadeiro) = 1$), a probabilidade de pessoas suspeitas é aumentada: $P(PessoasSuspeitas = verdadeiro|Ruido = falso) = 0.74216638$. Entretanto, o desejo $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$ ainda falha em satisfazer nosso limiar mesmo assim ($0.74216638 < 0.75$). Um limiar mais baixo funcionaria neste caso, mas a probabilidade de uma pré-condição pode ser sempre menor que o limiar, se evidência capaz de aumentar sua probabilidade suficientemente nunca for obtida – isto exemplifica como pode haver desejos que *nunca são* satisfeitos utilizando-se esse critério. Alguém poderia sugerir simplesmente diminuir o limiar para um valor extremamente baixo, mas sem outros critérios nós então simplesmente teríamos um processo de seleção de desejos indiferente às varias probabilidades apresentadas.

Se usamos seleção por Ranking Probabilístico, obtemos o seguinte ranking:

1. $Rota.padrao(Invasao.falso)$: 0.927
2. $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$: 0.7
3. $Rota.alternativa(Invasao.verdadeiro)$: 0.073
4. $Acidente.verdadeiro(Ruido.verdadeiro)$: 0.06241525

O agente desejará patrulhar a rota padrão, então apurar que não há falha elétrica, e finalmente patrulhar a rota alternativa, nesta ordem a não ser que uma atualização de crenças (e.g., evidência que $Invasao = verdadeiro$) faça com o que o ranking seja modificado. Observe que apesar de que as probabilidades pré-condicionantes servem como um critério para ordenar os desejos, os valores de probabilidade propriamente ditos não impactam o quão frequentemente os desejos podem ser selecionados, então $Rota.alternativa(Invasao.verdadeiro)$ será selecionado prontamente na ausência de desejos melhor rankeados independentemente do fato de que sua pré-condição possui uma probabilidade baixa.

Se usamos Loteria Viciada, obtemos a seguinte lista de intervalos numéricos para os desejos (a ordem é irrelevante):

- $Rota.padrao(Invasao.falso)$: $[0.0, 0.52598274)$
- $Rota.alternativa(Invasao.verdadeiro)$: $[0.52598274, 0.56740317)$
- $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$: $[0.56740317, 0.96458539)$
- $Acidente.verdadeiro(Ruido.verdadeiro)$: $[0.96458539, 1.0]$

O somatório das probabilidades de pré-condições dos desejos é maior que 1, então estes valores são normalizados no intervalo $[0, 1]$ e usados para gerar os intervalos. Seguindo o algoritmo, um valor numérico no intervalo $[0, 1]$ é gerado, e seja qual for o intervalo a que ele pertencer define qual desejo é selecionado – se não houvesse uma normalização, ele também poderia nos dizer que nenhum desejo deve ser selecionado, ao não pertencer a nenhum dos intervalos. Neste exemplo, o desejo $Rota.padrao(Invasao.falso)$ possui uma probabilidade de 0.52598274 de ser selecionado, o desejo $Rota.alternativa(Invasao.verdadeiro)$ possui uma probabilidade de 0.04142043 de ser selecionado, o desejo $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$ possui uma probabilidade de 0.39718222 de ser selecionado, e o desejo $Acidente.verdadeiro(Ruido.verdadeiro)$ possui uma probabilidade de 0.03541461 de ser selecionado, cada um competindo com os demais. Então, se o número aleatoriamente gerado é 0.3 (e portanto pertence ao primeiro intervalo), o agente realiza uma patrulha através da rota padrão, ou se o número aleatório é 0.55, a patrulha é através da rota alternativa.

Se usamos Seleção Aleatória Multidesejos com Viés, obtemos a seguinte lista de intervalos numéricos para os desejos (em qualquer ordem):

- $Rota.padrao(Invasao.falso)$: $[0.0, 0.927]$
- $Rota.alternativa(Invasao.verdadeiro)$: $[0.0, 0.073]$
- $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$: $[0.0, 0.7]$
- $Acidente.verdadeiro(Ruido.verdadeiro)$: $[0.0, 0.06241525]$

Para cada um dos quatro desejos um valor numérico no intervalo $[0, 1]$ é gerado, e se o valor numérico pertence ao intervalo numérico correspondente ao desejo, o desejo é selecionado. Neste exemplo, os desejos $Rota.padrao(Invasao.falso)$, $Rota.alternativa(Invasao.verdadeiro)$, $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$ e $Acidente.verdadeiro(Ruido.verdadeiro)$ possuem, respectivamente, probabilidades de 0.927, 0.073, 0.7 e 0.06241525 de serem selecionados ao receberem a chance (i.e., quando um número aleatório é gerado e checado contra o intervalo designado ao desejo). A seleção dos desejos $FalhaEletrica.falso(PessoasSuspeitas.verdadeiro)$ e $Acidente.verdadeiro(Ruido.verdadeiro)$ é totalmente independente das demais, enquanto os desejos $Rota.padrao(Invasao.falso)$ e $Rota.alternativa(Invasao.verdadeiro)$ conflitam um com o outro, e, em casos em que um deles é selecionado, isto impede que o outro receba sua chance.

Agora discutimos os algoritmos de seleção de desejos apresentados neste capítulo.

α5.5 Discussão

Dados os quatro algoritmos apresentados que podem ser usados para seleção de desejos em agentes BDI bayesianos, discutimos suas características.

De um ponto de vista conservador, alguém pode argumentar que seleção baseada em limiar é sensata como é, já que recursos não são usados *sem justificativa*. Entretanto, acreditamos que ignorar desejos que são probabilisticamente irrelevantes na seleção de desejos não é necessariamente uma decisão racional, já que isto impede que um agente explore o ambiente.

Em seleção por Ranking Probabilístico, os desejos que seriam ignorados por uma seleção baseada em limiar recebem uma chance, ainda que apenas após aqueles que seriam aceitos por ela. Todavia, pode ser indesejável selecionar um desejo pré-condicionado a uma crença associada a uma probabilidade muito baixa apenas porque não há alternativa melhor.

Em Loteria Viciada, contamos com não-determinismo, na forma computacionalmente viável de geração de números pseudoaleatórios, para considerar todos os desejos enquanto tendo em mente que desejos suportados por crenças associadas a altas probabilidades deveriam ser selecionadas mais frequentemente que aquelas suportadas por crenças associadas a probabilidades baixas, em proporção as suas probabilidades. Idealmente, a probabilidade de seleção de cada desejo seria a mesma que aquela associada a sua pré-condição, e isto é o que a pseudoaleatoriedade, juntamente a intervalos numéricos proporcionais aos valores probabilísticos envolvidos, busca emular. Entretanto, em casos em que o somatório de valores probabilísticos é maior que 1, a competição entre os desejos em questão diminui as probabilidades individuais de seleção para cada desejo. Não obstante, as proporções relativas entre as probabilidades para seleção de desejos são mantidas.

Em Seleção Aleatória Multidesejos com Viés, também contamos com não-determinismo, pela mesma razão. Uma diferença-chave é que o número de desejos possivelmente selecionados não é limitado a um, e que desejos não-conflitantes são considerados independentemente uns dos outros.

Em ambas Loteria Viciada e Seleção Aleatória Multidesejos com Viés, a qualidade do gerador de números pseudoaleatórios influencia o quão bem esses algoritmos emulam a probabilidade teórica de cada desejo ser selecionado. Sua natureza não-determinística faz com que o comportamento de agentes não possa ser previsto por terceiros (e.g., outro agente) com intenção de explorá-lo. No caso do agente vigia, tal exploração poderia envolver forjar um acidente para afastar suspeita oriunda de ruído percebido pelo vigia, por exemplo. Contratar um funcionário para plantar falsa evidência de falha elétrica também impactaria as crenças do vigia, como um segundo, ainda que mais indireto, método de tentativa de manipulação do vigia. Esta é uma característica de agentes que agora descrevemos como *proatividade imprevisível*: o comportamento de um agente em um momento específico não pode ser completamente determinado analisando-se suas crenças, sendo, portanto, resistente a explorações por terceiros.

Isso conclui nosso capítulo relacionado a abordagens para seleção de desejos BDI bayesianos. O Capítulo $\alpha 6$ apresenta nossas considerações finais.

α6 CONSIDERAÇÕES FINAIS

Nosso objetivo original era prover planejamento para agentes BDI que suportassem incerteza, dado conhecimento prévio de GRAPHPLAN em agentes BDI tradicionais; GRAPHPLAN (BLUM; FURST, 1997) é um algoritmo de planejamento rápido[1] que representa o fluxo de tempo através de uma estrutura de dados chamada *grafo de planejamento* – contendo níveis de proposições e de ações – e realiza uma busca encadeada para trás por um plano ótimo. Antes de percebermos que não podíamos subestimar a relevância ou desconsiderar Redes Bayesianas, investigamos duas extensões do GRAPHPLAN para lidar com planejamento suportando incerteza: PGRAPHPLAN e TGRAPHPLAN (BLUM; LANGFORD, 1998; LITTLE; THIÉBAUX, 2006). Esta preocupação com incerteza com o passar do tempo nos levou a Redes Bayesianas. Contudo, deparamonos com grande dificuldade, já que frequentemente encontramos literatura essencialmente dependente de formalismos teóricos. Uma motivação primária deste trabalho foi o estudo de Redes Bayesianas, o qual tomou a maior parte do esforço investido, utilizando fontes de conhecimento profundamente dependentes de formalismos e uma abordagem que não é algoritmicamente motivada.

Fagundes e Vicari apresentaram uma solução inicial para o processo de seleção de desejos (FAGUNDES; VICARI; COELHO, 2007) no seu modelo de agentes BDI bayesianos (FAGUNDES; VICARI, 2007), e nós apresentamos uma limitação e propusemos alternativas. Isto provou merecer mais atenção do que o antecipado, considerando que conhecimento detalhado necessário de Redes Bayesianas não foi simplesmente presumido, mas sim estudado e, consequentemente, apresentado nesta dissertação. Graças a isto, nosso objetivo original, Planejamento para Agentes BDI Bayesianos, não pôde ser perseguido.

No Capítulo α2, falamos sobre agentes BDI. Explicamos a estrutura principal de um agente BDI com respeito a seus componentes essenciais internos: crenças, desejos e intenções, assim como planos. A lógica de execução de um agente BDI tradicional também foi apresentada. Nós assinalamos que a ocorrência de mudanças no ambiente potencialmente impacta desejos quanto a sua validade – quanto a suas condições para seleção ainda estarem sendo satisfeitas – e que cegamente insistir em reavaliar os desejos neste sentido ou o oposto – ignorar o fato de ter havido mudanças – não é uma política ótima. De qualquer maneira, este é um cenário com o qual lidar de algum modo (e.g., uma ideia aparentemente boa envolveria decisões baseadas em estatísticas). Aquelas mudanças também podem impactar a viabilidade de planos correntes e mesmo suas intenções associadas, caso não haja planos possíveis para satisfazê-las – um cenário sondável em uma fase opcional de replanejamento. Planos voltados a satisfazer intenções podem ser

[1]Observe que planejamento por si só é um problema custoso (NEBEL, 2000).

gerados conforme necessário via algoritmos de planejamento específicos e/ou buscados em uma biblioteca de planos, que também pode conter planos manualmente definidos.

No Capítulo $\alpha 3$, inicialmente discutimos como a ciência da incerteza no raciocínio é relevante para se ampliar o espectro de problemas tratáveis, já que uma falta de informações desejáveis é de fato um componente de cenários realistas. Então nós apresentamos um pouco de Teoria da Probabilidade pertinente, incluindo probabilidades condicionais e cálculo probabilístico. Depois disto, cobrimos a *Regra de Bayes* – que serve como a fundação para Redes Bayesianas, e finalmente focamos nas Redes Bayesianas propriamente ditas. Os três tipos de conexão encontrados em Redes Bayesianas e a propriedade de *d-separação* foram apresentados, assim como outras propriedades e processos relevantes à criação e à atualização de Redes Bayesianas.

No Capítulo $\alpha 4$ apresentamos o conceito de Agentes BDI Bayesianos: agentes autônomos baseados no modelo de agente BDI, porém expandidos de modo a utilizarem Redes Bayesianas como o mecanismo de representação para crenças, acerca das quais desejos são definidos. Foi apresentado um possível ciclo de execução, assim como seus conceitos de *crenças*, *desejos* e *intenções*. Adicionalmente, propusemos os conceitos de *desejos fortes* e *desejos fracos*, a fim de considerar tanto a certeza quanto a incerteza ao avaliar se um desejo foi satisfeito.

No Capítulo $\alpha 5$ cobrimos abordagens para seleção de desejos para agentes BDI bayesianos. Foi explicado como uma característica de um método anterior falha em lidar com todos possíveis desejos – lidar com aqueles que são improváveis, porém possíveis. Visando passar por esta limitação, propusemos e detalhamos métodos alternativos de seleção de desejos: *Ranking Probabilístico*, *Loteria Viciada* e *Seleção Aleatória Multidesejos com Viés*. O primeiro seleciona desejos usando uma lista de desejos decrescentemente ordenada pelas probabilidades das pré-condições correspondentes, o segundo não-deterministicamente seleciona desejos, um de cada vez, de acordo com uma distribuição de probabilidades visando refletir as probabilidades previamente mencionadas, e o terceiro não-deterministicamente seleciona múltiplos desejos não-conflitantes de uma vez de acordo com estas probabilidades independentemente uns dos outros. Também apresentamos o exemplo de agente *Vigia* para ilustrar os algoritmos apresentados.

Possíveis objetivos para trabalhos futuros incluem a modelagem de agentes BDI bayesianos focando em cenários especificos, a avaliação dos algoritmos apresentados em tais cenários e usos conjuntos de Loteria Viciada e Seleção Multidesejos com Viés enquanto considerando possíveis incompatibilidades entre desejos. Cenários em jogos, e.g., invadir uma instalação protegida – onde os guardas são agentes – apresentam um caminho para experimentos aplicados. Adicionalmente, *planos* – incluindo, mas não limitados a planejamento – para agentes BDI bayesianos parecem dignos de investigação, e.g., planos em exemplos práticos modelados, assim como considerar *utilidade* e *custo* em seleção de desejos.

APPENDIX A WATCHMAN AGENT: SCENARIO PROBABILITIES

The probability of $SecurityBreach = true$ - and implicitly the probability of $SecurityBreach = false$ - varies considerably with the evidence available, as you can see in Table A.1. To a lesser degree, so does the probability of $SuspiciousPeople = true$. The table contains all the possible scenarios in terms of available evidence and the probabilities of $SecurityBreach = true$, $SuspiciousPeople = true$, $Alarm = true$ and $Noise = true$; evidence on $true$ states is denoted by t, on $false$ states by f and the lack thereof by $-$; also, ***SuspPeople*** denotes $SuspiciousPeople$, ***ElecMalfct*** denotes $ElectricalMalfunction$ and ***SecBreach*** denotes $SecurityBreach$.

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
–	–	–	–	–	–	0.073	0.7	0.098558	0.06241525
–	–	–	–	–	*t*	1.0	0.95890411	0.9755	0.1915
–	–	–	–	–	*f*	0.0	0.67961165	0.0295	0.05225
–	–	–	–	*t*	–	0.073	0.7	0.098558	0.95219
–	–	–	–	*t*	*t*	1.0	0.95890411	0.9755	0.98
–	–	–	–	*t*	*f*	0.0	0.67961165	0.0295	0.95
–	–	–	–	*f*	–	0.073	0.7	0.098558	0.015585
–	–	–	–	*f*	*t*	1.0	0.95890411	0.9755	0.15
–	–	–	–	*f*	*f*	0.0	0.67961165	0.0295	0.005
–	–	–	*t*	–	–	0.22397571	0.74216638	0.24138102	1.0
–	–	–	*t*	–	*t*	1.0	0.95890411	0.9755	1.0
–	–	–	*t*	–	*f*	0.0	0.67961165	0.0295	1.0
–	–	–	*t*	*t*	–	0.07513206	0.70059547	0.10057493	1.0
–	–	–	*t*	*t*	*t*	1.0	0.95890411	0.9755	1.0
–	–	–	*t*	*t*	*f*	0.0	0.67961165	0.0295	1.0
–	–	–	*t*	*f*	–	0.70259865	0.87584216	0.69415833	1.0
–	–	–	*t*	*f*	*t*	1.0	0.95890411	0.9755	1.0
–	–	–	*t*	*f*	*f*	0.0	0.67961165	0.0295	1.0
–	–	–	*f*	–	–	0.06294951	0.69719297	0.08905024	0.0
–	–	–	*f*	–	*t*	1.0	0.95890411	0.9755	0.0
–	–	–	*f*	–	*f*	0.0	0.67961165	0.0295	0.0
–	–	–	*f*	*t*	–	0.03053754	0.68814056	0.05838852	0.0
–	–	–	*f*	*t*	*t*	1.0	0.95890411	0.9755	0.0
–	–	–	*f*	*t*	*f*	0.0	0.67961165	0.0295	0.0
–	–	–	*f*	*f*	–	0.06303236	0.69721611	0.08912861	0.0
–	–	–	*f*	*f*	*t*	1.0	0.95890411	0.9755	0.0
–	–	–	*f*	*f*	*f*	0.0	0.67961165	0.0295	0.0
–	–	*t*	–	–	–	0.073	0.7	0.4219	0.06241525
–	–	*t*	–	–	*t*	1.0	0.95890411	0.7	0.1915
–	–	*t*	–	–	*f*	0.0	0.67961165	0.4	0.05225
–	–	*t*	–	*t*	–	0.073	0.7	0.4219	0.95219
–	–	*t*	–	*t*	*t*	1.0	0.95890411	0.7	0.98
–	–	*t*	–	*t*	*f*	0.0	0.67961165	0.4	0.95
–	–	*t*	–	*f*	–	0.073	0.7	0.4219	0.015585
–	–	*t*	–	*f*	*t*	1.0	0.95890411	0.7	0.15
–	–	*t*	–	*f*	*f*	0.0	0.67961165	0.4	0.005
–	–	*t*	*t*	–	–	0.22397571	0.74216638	0.46719271	1.0
–	–	*t*	*t*	–	*t*	1.0	0.95890411	0.7	1.0

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	P(SecBreach = true)	P(SuspPeople = true)	P(Alarm = true)	P(Noise = true)
–	–	*t*	*t*	–	*f*	0.0	0.67961165	0.4	1.0
–	–	*t*	*t*	*t*	–	0.07513206	0.70059547	0.42253962	1.0
–	–	*t*	*t*	*t*	*t*	1.0	0.95890411	0.7	1.0
–	–	*t*	*t*	*t*	*f*	0.0	0.67961165	0.4	1.0
–	–	*t*	*t*	*f*	–	0.70259865	0.87584216	0.6107796	1.0
–	–	*t*	*t*	*f*	*t*	1.0	0.95890411	0.7	1.0
–	–	*t*	*t*	*f*	*f*	0.0	0.67961165	0.4	1.0
–	–	*t*	*f*	–	–	0.06294951	0.69719297	0.41888485	0.0
–	–	*t*	*f*	–	*t*	1.0	0.95890411	0.7	0.0
–	–	*t*	*f*	–	*f*	0.0	0.67961165	0.4	0.0
–	–	*t*	*f*	*t*	–	0.03053754	0.68814056	0.40916126	0.0
–	–	*t*	*f*	*t*	*t*	1.0	0.95890411	0.7	0.0
–	–	*t*	*f*	*t*	*f*	0.0	0.67961165	0.4	0.0
–	–	*t*	*f*	*f*	–	0.06303236	0.69721611	0.41890971	0.0
–	–	*t*	*f*	*f*	*t*	1.0	0.95890411	0.7	0.0
–	–	*t*	*f*	*f*	*f*	0.0	0.67961165	0.4	0.0
–	–	*f*	–	–	–	0.073	0.7	0.08154	0.06241525
–	–	*f*	–	–	*t*	1.0	0.95890411	0.99	0.1915
–	–	*f*	–	–	*f*	0.0	0.67961165	0.01	0.05225
–	–	*f*	–	*t*	–	0.073	0.7	0.08154	0.95219
–	–	*f*	–	*t*	*t*	1.0	0.95890411	0.99	0.98
–	–	*f*	–	*t*	*f*	0.0	0.67961165	0.01	0.95
–	–	*f*	–	*f*	–	0.073	0.7	0.08154	0.015585
–	–	*f*	–	*f*	*t*	1.0	0.95890411	0.99	0.15
–	–	*f*	–	*f*	*f*	0.0	0.67961165	0.01	0.005
–	–	*f*	*t*	–	–	0.22397571	0.74216638	0.2294962	1.0
–	–	*f*	*t*	–	*t*	1.0	0.95890411	0.99	1.0
–	–	*f*	*t*	–	*f*	0.0	0.67961165	0.01	1.0
–	–	*f*	*t*	*t*	–	0.07513206	0.70059547	0.08362942	1.0
–	–	*f*	*t*	*t*	*t*	1.0	0.95890411	0.99	1.0
–	–	*f*	*t*	*t*	*f*	0.0	0.67961165	0.01	1.0
–	–	*f*	*t*	*f*	–	0.70259865	0.87584216	0.69854668	1.0
–	–	*f*	*t*	*f*	*t*	1.0	0.95890411	0.99	1.0
–	–	*f*	*t*	*f*	*f*	0.0	0.67961165	0.01	1.0
–	–	*f*	*f*	–	–	0.06294951	0.69719297	0.07169052	0.0
–	–	*f*	*f*	–	*t*	1.0	0.95890411	0.99	0.0
–	–	*f*	*f*	–	*f*	0.0	0.67961165	0.01	0.0
–	–	*f*	*f*	*t*	–	0.03053754	0.68814056	0.03992679	0.0
–	–	*f*	*f*	*t*	*t*	1.0	0.95890411	0.99	0.0
–	–	*f*	*f*	*t*	*f*	0.0	0.67961165	0.01	0.0
–	–	*f*	*f*	*f*	–	0.06303236	0.69721611	0.07177171	0.0
–	–	*f*	*f*	*f*	*t*	1.0	0.95890411	0.99	0.0
–	–	*f*	*f*	*f*	*f*	0.0	0.67961165	0.01	0.0
–	*t*	–	–	–	–	0.72253394	0.88140993	1.0	0.15286285
–	*t*	–	–	–	*t*	1.0	0.95890411	1.0	0.1915
–	*t*	–	–	–	*f*	0.0	0.67961165	1.0	0.05225
–	*t*	–	–	*t*	–	0.72253394	0.88140993	1.0	0.97167602
–	*t*	–	–	*t*	*t*	1.0	0.95890411	1.0	0.98
–	*t*	–	–	*t*	*f*	0.0	0.67961165	1.0	0.95
–	*t*	–	–	*f*	–	0.72253394	0.88140993	1.0	0.10976742
–	*t*	–	–	*f*	*t*	1.0	0.95890411	1.0	0.15
–	*t*	–	–	*f*	*f*	0.0	0.67961165	1.0	0.005
–	*t*	–	*t*	–	–	0.90515942	0.93241585	1.0	1.0
–	*t*	–	*t*	–	*t*	1.0	0.95890411	1.0	1.0
–	*t*	–	*t*	–	*f*	0.0	0.67961165	1.0	1.0
–	*t*	–	*t*	*t*	–	0.72872362	0.88313866	1.0	1.0
–	*t*	–	*t*	*t*	*t*	1.0	0.95890411	1.0	1.0
–	*t*	–	*t*	*t*	*f*	0.0	0.67961165	1.0	1.0
–	*t*	–	*t*	*f*	–	0.98736118	0.95537418	1.0	1.0
–	*t*	–	*t*	*f*	*t*	1.0	0.95890411	1.0	1.0
–	*t*	–	*t*	*f*	*f*	0.0	0.67961165	1.0	1.0
–	*t*	–	*f*	–	–	0.68957983	0.8722061	1.0	0.0
–	*t*	–	*f*	–	*t*	1.0	0.95890411	1.0	0.0
–	*t*	–	*f*	–	*f*	0.0	0.67961165	1.0	0.0
–	*t*	–	*f*	*t*	–	0.51019235	0.82210453	1.0	0.0
–	*t*	–	*f*	*t*	*t*	1.0	0.95890411	1.0	0.0
–	*t*	–	*f*	*t*	*f*	0.0	0.67961165	1.0	0.0
–	*t*	–	*f*	*f*	–	0.68988022	0.87228999	1.0	0.0
–	*t*	–	*f*	*f*	*t*	1.0	0.95890411	1.0	0.0
–	*t*	–	*f*	*f*	*f*	0.0	0.67961165	1.0	0.0
–	*t*	*t*	–	–	–	0.12111875	0.7134392	1.0	0.06911579

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
–	*t*	*t*	–	–	*t*	1.0	0.95890411	1.0	0.1915
–	*t*	*t*	–	–	*f*	0.0	0.67961165	1.0	0.05225
–	*t*	*t*	–	*t*	–	0.12111875	0.7134392	1.0	0.95363356
–	*t*	*t*	–	*t*	*t*	1.0	0.95890411	1.0	0.98
–	*t*	*t*	–	*t*	*f*	0.0	0.67961165	1.0	0.95
–	*t*	*t*	–	*f*	–	0.12111875	0.7134392	1.0	0.02256222
–	*t*	*t*	–	*f*	*t*	1.0	0.95890411	1.0	0.15
–	*t*	*t*	–	*f*	*f*	0.0	0.67961165	1.0	0.005
–	*t*	*t*	*t*	–	–	0.33558528	0.77333809	1.0	1.0
–	*t*	*t*	*t*	–	*t*	1.0	0.95890411	1.0	1.0
–	*t*	*t*	*t*	–	*f*	0.0	0.67961165	1.0	1.0
–	*t*	*t*	*t*	*t*	–	0.12446749	0.71437448	1.0	1.0
–	*t*	*t*	*t*	*t*	*t*	1.0	0.95890411	1.0	1.0
–	*t*	*t*	*t*	*t*	*f*	0.0	0.67961165	1.0	1.0
–	*t*	*t*	*t*	*f*	–	0.80523164	0.90450678	1.0	1.0
–	*t*	*t*	*t*	*f*	*t*	1.0	0.95890411	1.0	1.0
–	*t*	*t*	*t*	*f*	*f*	0.0	0.67961165	1.0	1.0
–	*t*	*t*	*f*	–	–	0.10519515	0.70899186	1.0	0.0
–	*t*	*t*	*f*	–	*t*	1.0	0.95890411	1.0	0.0
–	*t*	*t*	*f*	–	*f*	0.0	0.67961165	1.0	0.0
–	*t*	*t*	*f*	*t*	–	0.05224415	0.69420305	1.0	0.0
–	*t*	*t*	*f*	*t*	*t*	1.0	0.95890411	1.0	0.0
–	*t*	*t*	*f*	*t*	*f*	0.0	0.67961165	1.0	0.0
–	*t*	*t*	*f*	*f*	–	0.10532736	0.70902879	1.0	0.0
–	*t*	*t*	*f*	*f*	*t*	1.0	0.95890411	1.0	0.0
–	*t*	*t*	*f*	*f*	*f*	0.0	0.67961165	1.0	0.0
–	*t*	*f*	–	–	–	0.88631347	0.92715232	1.0	0.17566915
–	*t*	*f*	–	–	*t*	1.0	0.95890411	1.0	0.1915
–	*t*	*f*	–	–	*f*	0.0	0.67961165	1.0	0.05225
–	*t*	*f*	–	*t*	–	0.88631347	0.92715232	1.0	0.9765894
–	*t*	*f*	–	*t*	*t*	1.0	0.95890411	1.0	0.98
–	*t*	*f*	–	*t*	*f*	0.0	0.67961165	1.0	0.95
–	*t*	*f*	–	*f*	–	0.88631347	0.92715232	1.0	0.13351545
–	*t*	*f*	–	*f*	*t*	1.0	0.95890411	1.0	0.15
–	*t*	*f*	–	*f*	*f*	0.0	0.67961165	1.0	0.005
–	*t*	*f*	*t*	–	–	0.96618575	0.94946004	1.0	1.0
–	*t*	*f*	*t*	–	*t*	1.0	0.95890411	1.0	1.0
–	*t*	*f*	*t*	–	*f*	0.0	0.67961165	1.0	1.0
–	*t*	*f*	*t*	*t*	–	0.88940879	0.92801682	1.0	1.0
–	*t*	*f*	*t*	*t*	*t*	1.0	0.95890411	1.0	1.0
–	*t*	*f*	*t*	*t*	*f*	0.0	0.67961165	1.0	1.0
–	*t*	*f*	*t*	*f*	–	0.99574257	0.95771504	1.0	1.0
–	*t*	*f*	*t*	*f*	*t*	1.0	0.95890411	1.0	1.0
–	*t*	*f*	*t*	*f*	*f*	0.0	0.67961165	1.0	1.0
–	*t*	*f*	*f*	–	–	0.86929227	0.92239843	1.0	0.0
–	*t*	*f*	*f*	–	*t*	1.0	0.95890411	1.0	0.0
–	*t*	*f*	*f*	–	*f*	0.0	0.67961165	1.0	0.0
–	*t*	*f*	*f*	*t*	–	0.75719	0.89108911	1.0	0.0
–	*t*	*f*	*f*	*t*	*t*	1.0	0.95890411	1.0	0.0
–	*t*	*f*	*f*	*t*	*f*	0.0	0.67961165	1.0	0.0
–	*t*	*f*	*f*	*f*	–	0.86945168	0.92244295	1.0	0.0
–	*t*	*f*	*f*	*f*	*t*	1.0	0.95890411	1.0	0.0
–	*t*	*f*	*f*	*f*	*f*	0.0	0.67961165	1.0	0.0
–	*f*	–	–	–	–	0.00198404	0.68016578	0.0	0.05252628
–	*f*	–	–	–	*t*	1.0	0.95890411	0.0	0.1915
–	*f*	–	–	–	*f*	0.0	0.67961165	0.0	0.05225
–	*f*	–	–	*t*	–	0.00198404	0.68016578	0.0	0.95005952
–	*f*	–	–	*t*	*t*	1.0	0.95890411	0.0	0.98
–	*f*	–	–	*t*	*f*	0.0	0.67961165	0.0	0.95
–	*f*	–	–	*f*	–	0.00198404	0.68016578	0.0	0.00528769
–	*f*	–	–	*f*	*t*	1.0	0.95890411	0.0	0.15
–	*f*	–	–	*f*	*f*	0.0	0.67961165	0.0	0.005
–	*f*	–	*t*	–	–	0.00723341	0.68163189	0.0	1.0
–	*f*	–	*t*	–	*t*	1.0	0.95890411	0.0	1.0
–	*f*	–	*t*	–	*f*	0.0	0.67961165	0.0	1.0
–	*f*	–	*t*	*t*	–	0.00204657	0.68018324	0.0	1.0
–	*f*	–	*t*	*t*	*t*	1.0	0.95890411	0.0	1.0
–	*f*	–	*t*	*t*	*f*	0.0	0.67961165	0.0	1.0
–	*f*	–	*t*	*f*	–	0.05628293	0.69533105	0.0	1.0
–	*f*	–	*t*	*f*	*t*	1.0	0.95890411	0.0	1.0
–	*f*	–	*t*	*f*	*f*	0.0	0.67961165	0.0	1.0

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
–	*f*	–	*f*	–	–	0.00169303	0.6800845	0.0	0.0
–	*f*	–	*f*	–	*t*	1.0	0.95890411	0.0	0.0
–	*f*	–	*f*	–	*f*	0.0	0.67961165	0.0	0.0
–	*f*	–	*f*	*t*	–	0.00079456	0.67983357	0.0	0.0
–	*f*	–	*f*	*t*	*t*	1.0	0.95890411	0.0	0.0
–	*f*	–	*f*	*t*	*f*	0.0	0.67961165	0.0	0.0
–	*f*	–	*f*	*f*	–	0.0016954	0.68008516	0.0	0.0
–	*f*	–	*f*	*f*	*t*	1.0	0.95890411	0.0	0.0
–	*f*	–	*f*	*f*	*f*	0.0	0.67961165	0.0	0.0
–	*f*	*t*	–	–	–	0.03788272	0.69019201	0.0	0.05752517
–	*f*	*t*	–	–	*t*	1.0	0.95890411	0.0	0.1915
–	*f*	*t*	–	–	*f*	0.0	0.67961165	0.0	0.05225
–	*f*	*t*	–	*t*	–	0.03788272	0.69019201	0.0	0.95113648
–	*f*	*t*	–	*t*	*t*	1.0	0.95890411	0.0	0.98
–	*f*	*t*	–	*t*	*f*	0.0	0.67961165	0.0	0.95
–	*f*	*t*	–	*f*	–	0.03788272	0.69019201	0.0	0.01049299
–	*f*	*t*	–	*f*	*t*	1.0	0.95890411	0.0	0.15
–	*f*	*t*	–	*f*	*f*	0.0	0.67961165	0.0	0.005
–	*f*	*t*	*t*	–	–	0.12611073	0.71483343	0.0	1.0
–	*f*	*t*	*t*	–	*t*	1.0	0.95890411	0.0	1.0
–	*f*	*t*	*t*	–	*f*	0.0	0.67961165	0.0	1.0
–	*f*	*t*	*t*	*t*	–	0.03903232	0.69051308	0.0	1.0
–	*f*	*t*	*t*	*t*	*t*	1.0	0.95890411	0.0	1.0
–	*f*	*t*	*t*	*t*	*f*	0.0	0.67961165	0.0	1.0
–	*f*	*t*	*t*	*f*	–	0.54154303	0.83086053	0.0	1.0
–	*f*	*t*	*t*	*f*	*t*	1.0	0.95890411	0.0	1.0
–	*f*	*t*	*t*	*f*	*f*	0.0	0.67961165	0.0	1.0
–	*f*	*t*	*f*	–	–	0.03249761	0.68868799	0.0	0.0
–	*f*	*t*	*f*	–	*t*	1.0	0.95890411	0.0	0.0
–	*f*	*t*	*f*	–	*f*	0.0	0.67961165	0.0	0.0
–	*f*	*t*	*f*	*t*	–	0.01550552	0.68394223	0.0	0.0
–	*f*	*t*	*f*	*t*	*t*	1.0	0.95890411	0.0	0.0
–	*f*	*t*	*f*	*t*	*f*	0.0	0.67961165	0.0	0.0
–	*f*	*t*	*f*	*f*	–	0.03254177	0.68870032	0.0	0.0
–	*f*	*t*	*f*	*f*	*t*	1.0	0.95890411	0.0	0.0
–	*f*	*t*	*f*	*f*	*f*	0.0	0.67961165	0.0	0.0
–	*f*	*f*	–	–	–	0.00079481	0.67983363	0.0	0.05236068
–	*f*	*f*	–	–	*t*	1.0	0.95890411	0.0	0.1915
–	*f*	*f*	–	–	*f*	0.0	0.67961165	0.0	0.05225
–	*f*	*f*	–	*t*	–	0.00079481	0.67983363	0.0	0.95002384
–	*f*	*f*	–	*t*	*t*	1.0	0.95890411	0.0	0.98
–	*f*	*f*	–	*t*	*f*	0.0	0.67961165	0.0	0.95
–	*f*	*f*	–	*f*	–	0.00079481	0.67983363	0.0	0.00511525
–	*f*	*f*	–	*f*	*t*	1.0	0.95890411	0.0	0.15
–	*f*	*f*	–	*f*	*f*	0.0	0.67961165	0.0	0.005
–	*f*	*f*	*t*	–	–	0.00290687	0.68042352	0.0	1.0
–	*f*	*f*	*t*	–	*t*	1.0	0.95890411	0.0	1.0
–	*f*	*f*	*t*	–	*f*	0.0	0.67961165	0.0	1.0
–	*f*	*f*	*t*	*t*	–	0.00081989	0.67984064	0.0	1.0
–	*f*	*f*	*t*	*t*	*t*	1.0	0.95890411	0.0	1.0
–	*f*	*f*	*t*	*t*	*f*	0.0	0.67961165	0.0	1.0
–	*f*	*f*	*t*	*f*	–	0.02330705	0.68612113	0.0	1.0
–	*f*	*f*	*t*	*f*	*t*	1.0	0.95890411	0.0	1.0
–	*f*	*f*	*t*	*f*	*f*	0.0	0.67961165	0.0	1.0
–	*f*	*f*	*f*	–	–	0.00067811	0.67980104	0.0	0.0
–	*f*	*f*	*f*	–	*t*	1.0	0.95890411	0.0	0.0
–	*f*	*f*	*f*	–	*f*	0.0	0.67961165	0.0	0.0
–	*f*	*f*	*f*	*t*	–	0.00031808	0.67970049	0.0	0.0
–	*f*	*f*	*f*	*t*	*t*	1.0	0.95890411	0.0	0.0
–	*f*	*f*	*f*	*t*	*f*	0.0	0.67961165	0.0	0.0
–	*f*	*f*	*f*	*f*	–	0.00067906	0.67980131	0.0	0.0
–	*f*	*f*	*f*	*f*	*t*	1.0	0.95890411	0.0	0.0
–	*f*	*f*	*f*	*f*	*f*	0.0	0.67961165	0.0	0.0
t	–	–	–	–	–	0.1	1.0	0.1241	0.066175
t	–	–	–	–	*t*	1.0	1.0	0.9755	0.1915
t	–	–	–	–	*f*	0.0	1.0	0.0295	0.05225
t	–	–	–	*t*	–	0.1	1.0	0.1241	0.953
t	–	–	–	*t*	*t*	1.0	1.0	0.9755	0.98
t	–	–	–	*t*	*f*	0.0	1.0	0.0295	0.95
t	–	–	–	*f*	–	0.1	1.0	0.1241	0.0195
t	–	–	–	*f*	*t*	1.0	1.0	0.9755	0.15

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
t	–	–	–	*f*	*f*	0.0	1.0	0.0295	0.005
t	–	–	*t*	–	–	0.28938421	1.0	0.30325746	1.0
t	–	–	*t*	–	*t*	1.0	1.0	0.9755	1.0
t	–	–	*t*	–	*f*	0.0	1.0	0.0295	1.0
t	–	–	*t*	*t*	–	0.10283316	1.0	0.12678017	1.0
t	–	–	*t*	*t*	*t*	1.0	1.0	0.9755	1.0
t	–	–	*t*	*t*	*f*	0.0	1.0	0.0295	1.0
t	–	–	*t*	*f*	–	0.76923077	1.0	0.75719231	1.0
t	–	–	*t*	*f*	*t*	1.0	1.0	0.9755	1.0
t	–	–	*t*	*f*	*f*	0.0	1.0	0.0295	1.0
t	–	–	*f*	–	–	0.08657939	1.0	0.1114041	0.0
t	–	–	*f*	–	*t*	1.0	1.0	0.9755	0.0
t	–	–	*f*	–	*f*	0.0	1.0	0.0295	0.0
t	–	–	*f*	*t*	–	0.04255319	1.0	0.06975532	0.0
t	–	–	*f*	*t*	*t*	1.0	1.0	0.9755	0.0
t	–	–	*f*	*t*	*f*	0.0	1.0	0.0295	0.0
t	–	–	*f*	*f*	–	0.08669046	1.0	0.11150918	0.0
t	–	–	*f*	*f*	*t*	1.0	1.0	0.9755	0.0
t	–	–	*f*	*f*	*f*	0.0	1.0	0.0295	0.0
t	–	*t*	–	–	–	0.1	1.0	0.43	0.066175
t	–	*t*	–	–	*t*	1.0	1.0	0.7	0.1915
t	–	*t*	–	–	*f*	0.0	1.0	0.4	0.05225
t	–	*t*	–	*t*	–	0.1	1.0	0.43	0.953
t	–	*t*	–	*t*	*t*	1.0	1.0	0.7	0.98
t	–	*t*	–	*t*	*f*	0.0	1.0	0.4	0.95
t	–	*t*	–	*f*	–	0.1	1.0	0.43	0.0195
t	–	*t*	–	*f*	*t*	1.0	1.0	0.7	0.15
t	–	*t*	–	*f*	*f*	0.0	1.0	0.4	0.005
t	–	*t*	*t*	–	–	0.28938421	1.0	0.48681526	1.0
t	–	*t*	*t*	–	*t*	1.0	1.0	0.7	1.0
t	–	*t*	*t*	–	*f*	0.0	1.0	0.4	1.0
t	–	*t*	*t*	*t*	–	0.10283316	1.0	0.43084995	1.0
t	–	*t*	*t*	*t*	*t*	1.0	1.0	0.7	1.0
t	–	*t*	*t*	*t*	*f*	0.0	1.0	0.4	1.0
t	–	*t*	*t*	*f*	–	0.76923077	1.0	0.63076923	1.0
t	–	*t*	*t*	*f*	*t*	1.0	1.0	0.7	1.0
t	–	*t*	*t*	*f*	*f*	0.0	1.0	0.4	1.0
t	–	*t*	*f*	–	–	0.08657939	1.0	0.42597382	0.0
t	–	*t*	*f*	–	*t*	1.0	1.0	0.7	0.0
t	–	*t*	*f*	–	*f*	0.0	1.0	0.4	0.0
t	–	*t*	*f*	*t*	–	0.04255319	1.0	0.41276596	0.0
t	–	*t*	*f*	*t*	*t*	1.0	1.0	0.7	0.0
t	–	*t*	*f*	*t*	*f*	0.0	1.0	0.4	0.0
t	–	*t*	*f*	*f*	–	0.08669046	1.0	0.42600714	0.0
t	–	*t*	*f*	*f*	*t*	1.0	1.0	0.7	0.0
t	–	*t*	*f*	*f*	*f*	0.0	1.0	0.4	0.0
t	–	*f*	–	–	–	0.1	1.0	0.108	0.066175
t	–	*f*	–	–	*t*	1.0	1.0	0.99	0.1915
t	–	*f*	–	–	*f*	0.0	1.0	0.01	0.05225
t	–	*f*	–	*t*	–	0.1	1.0	0.108	0.953
t	–	*f*	–	*t*	*t*	1.0	1.0	0.99	0.98
t	–	*f*	–	*t*	*f*	0.0	1.0	0.01	0.95
t	–	*f*	–	*f*	–	0.1	1.0	0.108	0.0195
t	–	*f*	–	*f*	*t*	1.0	1.0	0.99	0.15
t	–	*f*	–	*f*	*f*	0.0	1.0	0.01	0.005
t	–	*f*	*t*	–	–	0.28938421	1.0	0.29359652	1.0
t	–	*f*	*t*	–	*t*	1.0	1.0	0.99	1.0
t	–	*f*	*t*	–	*f*	0.0	1.0	0.01	1.0
t	–	*f*	*t*	*t*	–	0.10283316	1.0	0.1107765	1.0
t	–	*f*	*t*	*t*	*t*	1.0	1.0	0.99	1.0
t	–	*f*	*t*	*t*	*f*	0.0	1.0	0.01	1.0
t	–	*f*	*t*	*f*	–	0.76923077	1.0	0.76384615	1.0
t	–	*f*	*t*	*f*	*t*	1.0	1.0	0.99	1.0
t	–	*f*	*t*	*f*	*f*	0.0	1.0	0.01	1.0
t	–	*f*	*f*	–	–	0.08657939	1.0	0.0948478	0.0
t	–	*f*	*f*	–	*t*	1.0	1.0	0.99	0.0
t	–	*f*	*f*	–	*f*	0.0	1.0	0.01	0.0
t	–	*f*	*f*	*t*	–	0.04255319	1.0	0.05170213	0.0
t	–	*f*	*f*	*t*	*t*	1.0	1.0	0.99	0.0
t	–	*f*	*f*	*t*	*f*	0.0	1.0	0.01	0.0
t	–	*f*	*f*	*f*	–	0.08669046	1.0	0.09495665	0.0

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
t	–	*f*	*f*	*f*	*t*	1.0	1.0	0.99	0.0
t	–	*f*	*f*	*f*	*f*	0.0	1.0	0.01	0.0
t	*t*	–	–	–	–	0.78605963	1.0	1.0	0.1617088
t	*t*	–	–	–	*t*	1.0	1.0	1.0	0.1915
t	*t*	–	–	–	*f*	0.0	1.0	1.0	0.05225
t	*t*	–	–	*t*	–	0.78605963	1.0	1.0	0.97358179
t	*t*	–	–	*t*	*t*	1.0	1.0	1.0	0.98
t	*t*	–	–	*t*	*f*	0.0	1.0	1.0	0.95
t	*t*	–	–	*f*	–	0.78605963	1.0	1.0	0.11897865
t	*t*	–	–	*f*	*t*	1.0	1.0	1.0	0.15
t	*t*	–	–	*f*	*f*	0.0	1.0	1.0	0.005
t	*t*	–	*t*	–	–	0.93087337	1.0	1.0	1.0
t	*t*	–	*t*	–	*t*	1.0	1.0	1.0	1.0
t	*t*	–	*t*	–	*f*	0.0	1.0	1.0	1.0
t	*t*	–	*t*	*t*	–	0.79124163	1.0	1.0	1.0
t	*t*	–	*t*	*t*	*t*	1.0	1.0	1.0	1.0
t	*t*	–	*t*	*t*	*f*	0.0	1.0	1.0	1.0
t	*t*	–	*t*	*f*	–	0.9910093	1.0	1.0	1.0
t	*t*	–	*t*	*f*	*t*	1.0	1.0	1.0	1.0
t	*t*	–	*t*	*f*	*f*	0.0	1.0	1.0	1.0
t	*t*	–	*f*	–	–	0.75812464	1.0	1.0	0.0
t	*t*	–	*f*	–	*t*	1.0	1.0	1.0	0.0
t	*t*	–	*f*	–	*f*	0.0	1.0	1.0	0.0
t	*t*	–	*f*	*t*	–	0.59508922	1.0	1.0	0.0
t	*t*	–	*f*	*t*	*t*	1.0	1.0	1.0	0.0
t	*t*	–	*f*	*t*	*f*	0.0	1.0	1.0	0.0
t	*t*	–	*f*	*f*	–	0.75838194	1.0	1.0	0.0
t	*t*	–	*f*	*f*	*t*	1.0	1.0	1.0	0.0
t	*t*	–	*f*	*f*	*f*	0.0	1.0	1.0	0.0
t	*t*	*t*	–	–	–	0.1627907	1.0	1.0	0.0749186
t	*t*	*t*	–	–	*t*	1.0	1.0	1.0	0.1915
t	*t*	*t*	–	–	*f*	0.0	1.0	1.0	0.05225
t	*t*	*t*	–	*t*	–	0.1627907	1.0	1.0	0.95488372
t	*t*	*t*	–	*t*	*t*	1.0	1.0	1.0	0.98
t	*t*	*t*	–	*t*	*f*	0.0	1.0	1.0	0.95
t	*t*	*t*	–	*f*	–	0.1627907	1.0	1.0	0.02860465
t	*t*	*t*	–	*f*	*t*	1.0	1.0	1.0	0.15
t	*t*	*t*	–	*f*	*f*	0.0	1.0	1.0	0.005
t	*t*	*t*	*t*	–	–	0.41611051	1.0	1.0	1.0
t	*t*	*t*	*t*	–	*t*	1.0	1.0	1.0	1.0
t	*t*	*t*	*t*	–	*f*	0.0	1.0	1.0	1.0
t	*t*	*t*	*t*	*t*	–	0.16707258	1.0	1.0	1.0
t	*t*	*t*	*t*	*t*	*t*	1.0	1.0	1.0	1.0
t	*t*	*t*	*t*	*t*	*f*	0.0	1.0	1.0	1.0
t	*t*	*t*	*t*	*f*	–	0.85365854	1.0	1.0	1.0
t	*t*	*t*	*t*	*f*	*t*	1.0	1.0	1.0	1.0
t	*t*	*t*	*t*	*f*	*f*	0.0	1.0	1.0	1.0
t	*t*	*t*	*f*	–	–	0.14227535	1.0	1.0	0.0
t	*t*	*t*	*f*	–	*t*	1.0	1.0	1.0	0.0
t	*t*	*t*	*f*	–	*f*	0.0	1.0	1.0	0.0
t	*t*	*t*	*f*	*t*	–	0.07216495	1.0	1.0	0.0
t	*t*	*t*	*f*	*t*	*t*	1.0	1.0	1.0	0.0
t	*t*	*t*	*f*	*t*	*f*	0.0	1.0	1.0	0.0
t	*t*	*t*	*f*	*f*	–	0.14244673	1.0	1.0	0.0
t	*t*	*t*	*f*	*f*	*t*	1.0	1.0	1.0	0.0
t	*t*	*t*	*f*	*f*	*f*	0.0	1.0	1.0	0.0
t	*t*	*f*	–	–	–	0.91666667	1.0	1.0	0.17989583
t	*t*	*f*	–	–	*t*	1.0	1.0	1.0	0.1915
t	*t*	*f*	–	–	*f*	0.0	1.0	1.0	0.05225
t	*t*	*f*	–	*t*	–	0.91666667	1.0	1.0	0.9775
t	*t*	*f*	–	*t*	*t*	1.0	1.0	1.0	0.98
t	*t*	*f*	–	*t*	*f*	0.0	1.0	1.0	0.95
t	*t*	*f*	–	*f*	–	0.91666667	1.0	1.0	0.13791667
t	*t*	*f*	–	*f*	*t*	1.0	1.0	1.0	0.15
t	*t*	*f*	–	*f*	*f*	0.0	1.0	1.0	0.005
t	*t*	*f*	*t*	–	–	0.97579618	1.0	1.0	1.0
t	*t*	*f*	*t*	–	*t*	1.0	1.0	1.0	1.0
t	*t*	*f*	*t*	–	*f*	0.0	1.0	1.0	1.0
t	*t*	*f*	*t*	*t*	–	0.91901108	1.0	1.0	1.0
t	*t*	*f*	*t*	*t*	*t*	1.0	1.0	1.0	1.0
t	*t*	*f*	*t*	*t*	*f*	0.0	1.0	1.0	1.0

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
t	*t*	*f*	*t*	*f*	–	0.99697885	1.0	1.0	1.0
t	*t*	*f*	*t*	*f*	*t*	1.0	1.0	1.0	1.0
t	*t*	*f*	*t*	*f*	*f*	0.0	1.0	1.0	1.0
t	*t*	*f*	*f*	–	–	0.90369618	1.0	1.0	0.0
t	*t*	*f*	*f*	–	*t*	1.0	1.0	1.0	0.0
t	*t*	*f*	*f*	–	*f*	0.0	1.0	1.0	0.0
t	*t*	*f*	*f*	*t*	–	0.81481481	1.0	1.0	0.0
t	*t*	*f*	*f*	*t*	*t*	1.0	1.0	1.0	0.0
t	*t*	*f*	*f*	*t*	*f*	0.0	1.0	1.0	0.0
t	*t*	*f*	*f*	*f*	–	0.90381827	1.0	1.0	0.0
t	*t*	*f*	*f*	*f*	*t*	1.0	1.0	1.0	0.0
t	*t*	*f*	*f*	*f*	*f*	0.0	1.0	1.0	0.0
t	*f*	–	–	–	–	0.00279712	1.0	0.0	0.0526395
t	*f*	–	–	–	*t*	1.0	1.0	0.0	0.1915
t	*f*	–	–	–	*f*	0.0	1.0	0.0	0.05225
t	*f*	–	–	*t*	–	0.00279712	1.0	0.0	0.95008391
t	*f*	–	–	*t*	*t*	1.0	1.0	0.0	0.98
t	*f*	–	–	*t*	*f*	0.0	1.0	0.0	0.95
t	*f*	–	–	*f*	–	0.00279712	1.0	0.0	0.00540558
t	*f*	–	–	*f*	*t*	1.0	1.0	0.0	0.15
t	*f*	–	–	*f*	*f*	0.0	1.0	0.0	0.005
t	*f*	–	*t*	–	–	0.0101758	1.0	0.0	1.0
t	*f*	–	*t*	–	*t*	1.0	1.0	0.0	1.0
t	*f*	–	*t*	–	*f*	0.0	1.0	0.0	1.0
t	*f*	–	*t*	*t*	–	0.0028852	1.0	0.0	1.0
t	*f*	–	*t*	*t*	*t*	1.0	1.0	0.0	1.0
t	*f*	–	*t*	*t*	*f*	0.0	1.0	0.0	1.0
t	*f*	–	*t*	*f*	–	0.07761761	1.0	0.0	1.0
t	*f*	–	*t*	*f*	*t*	1.0	1.0	0.0	1.0
t	*f*	–	*t*	*f*	*f*	0.0	1.0	0.0	1.0
t	*f*	–	*f*	–	–	0.00238713	1.0	0.0	0.0
t	*f*	–	*f*	–	*t*	1.0	1.0	0.0	0.0
t	*f*	–	*f*	–	*f*	0.0	1.0	0.0	0.0
t	*f*	–	*f*	*t*	–	0.00112073	1.0	0.0	0.0
t	*f*	–	*f*	*t*	*t*	1.0	1.0	0.0	0.0
t	*f*	–	*f*	*t*	*f*	0.0	1.0	0.0	0.0
t	*f*	–	*f*	*f*	–	0.00239048	1.0	0.0	0.0
t	*f*	–	*f*	*f*	*t*	1.0	1.0	0.0	0.0
t	*f*	–	*f*	*f*	*f*	0.0	1.0	0.0	0.0
t	*f*	*t*	–	–	–	0.05263158	1.0	0.0	0.05957895
t	*f*	*t*	–	–	*t*	1.0	1.0	0.0	0.1915
t	*f*	*t*	–	–	*f*	0.0	1.0	0.0	0.05225
t	*f*	*t*	–	*t*	–	0.05263158	1.0	0.0	0.95157895
t	*f*	*t*	–	*t*	*t*	1.0	1.0	0.0	0.98
t	*f*	*t*	–	*t*	*f*	0.0	1.0	0.0	0.95
t	*f*	*t*	–	*f*	–	0.05263158	1.0	0.0	0.01263158
t	*f*	*t*	–	*f*	*t*	1.0	1.0	0.0	0.15
t	*f*	*t*	–	*f*	*f*	0.0	1.0	0.0	0.005
t	*f*	*t*	*t*	–	–	0.16916961	1.0	0.0	1.0
t	*f*	*t*	*t*	–	*t*	1.0	1.0	0.0	1.0
t	*f*	*t*	*t*	–	*f*	0.0	1.0	0.0	1.0
t	*f*	*t*	*t*	*t*	–	0.05420354	1.0	0.0	1.0
t	*f*	*t*	*t*	*t*	*t*	1.0	1.0	0.0	1.0
t	*f*	*t*	*t*	*t*	*f*	0.0	1.0	0.0	1.0
t	*f*	*t*	*t*	*f*	–	0.625	1.0	0.0	1.0
t	*f*	*t*	*t*	*f*	*t*	1.0	1.0	0.0	1.0
t	*f*	*t*	*t*	*f*	*f*	0.0	1.0	0.0	1.0
t	*f*	*t*	*f*	–	–	0.04524849	1.0	0.0	0.0
t	*f*	*t*	*f*	–	*t*	1.0	1.0	0.0	0.0
t	*f*	*t*	*f*	–	*f*	0.0	1.0	0.0	0.0
t	*f*	*t*	*f*	*t*	–	0.02173913	1.0	0.0	0.0
t	*f*	*t*	*f*	*t*	*t*	1.0	1.0	0.0	0.0
t	*f*	*t*	*f*	*t*	*f*	0.0	1.0	0.0	0.0
t	*f*	*t*	*f*	*f*	–	0.04530917	1.0	0.0	0.0
t	*f*	*t*	*f*	*f*	*t*	1.0	1.0	0.0	0.0
t	*f*	*t*	*f*	*f*	*f*	0.0	1.0	0.0	0.0
t	*f*	*f*	–	–	–	0.00112108	1.0	0.0	0.05240611
t	*f*	*f*	–	–	*t*	1.0	1.0	0.0	0.1915
t	*f*	*f*	–	–	*f*	0.0	1.0	0.0	0.05225
t	*f*	*f*	–	*t*	–	0.00112108	1.0	0.0	0.95003363
t	*f*	*f*	–	*t*	*t*	1.0	1.0	0.0	0.98

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
t	f	f	–	t	f	0.0	1.0	0.0	0.95
t	f	f	–	f	–	0.00112108	1.0	0.0	0.00516256
t	f	f	–	f	t	1.0	1.0	0.0	0.15
t	f	f	–	f	f	0.0	1.0	0.0	0.005
t	f	f	t	–	–	0.00409659	1.0	0.0	1.0
t	f	f	t	–	t	1.0	1.0	0.0	1.0
t	f	f	t	–	f	0.0	1.0	0.0	1.0
t	f	f	t	t	–	0.00115644	1.0	0.0	1.0
t	f	f	t	t	t	1.0	1.0	0.0	1.0
t	f	f	t	t	f	0.0	1.0	0.0	1.0
t	f	f	t	f	–	0.03257329	1.0	0.0	1.0
t	f	f	t	f	t	1.0	1.0	0.0	1.0
t	f	f	t	f	f	0.0	1.0	0.0	1.0
t	f	f	f	–	–	0.00095652	1.0	0.0	0.0
t	f	f	f	–	t	1.0	1.0	0.0	0.0
t	f	f	f	–	f	0.0	1.0	0.0	0.0
t	f	f	f	t	–	0.00044873	1.0	0.0	0.0
t	f	f	f	t	t	1.0	1.0	0.0	0.0
t	f	f	f	t	f	0.0	1.0	0.0	0.0
t	f	f	f	f	–	0.00095786	1.0	0.0	0.0
t	f	f	f	f	t	1.0	1.0	0.0	0.0
t	f	f	f	f	f	0.0	1.0	0.0	0.0
f	–	–	–	–	–	0.01	0.0	0.03896	0.0536425
f	–	–	–	–	t	1.0	0.0	0.9755	0.1915
f	–	–	–	–	f	0.0	0.0	0.0295	0.05225
f	–	–	–	t	–	0.01	0.0	0.03896	0.9503
f	–	–	–	t	t	1.0	0.0	0.9755	0.98
f	–	–	–	t	f	0.0	0.0	0.0295	0.95
f	–	–	–	f	–	0.01	0.0	0.03896	0.00645
f	–	–	–	f	t	1.0	0.0	0.9755	0.15
f	–	–	–	f	f	0.0	0.0	0.0295	0.005
f	–	–	t	–	–	0.03569931	0.0	0.06327154	1.0
f	–	–	t	–	t	1.0	0.0	0.9755	1.0
f	–	–	t	–	f	0.0	0.0	0.0295	1.0
f	–	–	t	t	–	0.01031253	0.0	0.03925566	1.0
f	–	–	t	t	t	1.0	0.0	0.9755	1.0
f	–	–	t	t	f	0.0	0.0	0.0295	1.0
f	–	–	t	f	–	0.23255814	0.0	0.2495	1.0
f	–	–	t	f	t	1.0	0.0	0.9755	1.0
f	–	–	t	f	f	0.0	0.0	0.0295	1.0
f	–	–	f	–	–	0.00854328	0.0	0.03758195	0.0
f	–	–	f	–	t	1.0	0.0	0.9755	0.0
f	–	–	f	–	f	0.0	0.0	0.0295	0.0
f	–	–	f	t	–	0.00402414	0.0	0.03330684	0.0
f	–	–	f	t	t	1.0	0.0	0.9755	0.0
f	–	–	f	t	f	0.0	0.0	0.0295	0.0
f	–	–	f	f	–	0.00855518	0.0	0.0375932	0.0
f	–	–	f	f	t	1.0	0.0	0.9755	0.0
f	–	–	f	f	f	0.0	0.0	0.0295	0.0
f	–	t	–	–	–	0.01	0.0	0.403	0.0536425
f	–	t	–	–	t	1.0	0.0	0.7	0.1915
f	–	t	–	–	f	0.0	0.0	0.4	0.05225
f	–	t	–	t	–	0.01	0.0	0.403	0.9503
f	–	t	–	t	t	1.0	0.0	0.7	0.98
f	–	t	–	t	f	0.0	0.0	0.4	0.95
f	–	t	–	f	–	0.01	0.0	0.403	0.00645
f	–	t	–	f	t	1.0	0.0	0.7	0.15
f	–	t	–	f	f	0.0	0.0	0.4	0.005
f	–	t	t	–	–	0.03569931	0.0	0.41070979	1.0
f	–	t	t	–	t	1.0	0.0	0.7	1.0
f	–	t	t	–	f	0.0	0.0	0.4	1.0
f	–	t	t	t	–	0.01031253	0.0	0.40309376	1.0
f	–	t	t	t	t	1.0	0.0	0.7	1.0
f	–	t	t	t	f	0.0	0.0	0.4	1.0
f	–	t	t	f	–	0.23255814	0.0	0.46976744	1.0
f	–	t	t	f	t	1.0	0.0	0.7	1.0
f	–	t	t	f	f	0.0	0.0	0.4	1.0
f	–	t	f	–	–	0.00854328	0.0	0.40256298	0.0
f	–	t	f	–	t	1.0	0.0	0.7	0.0
f	–	t	f	–	f	0.0	0.0	0.4	0.0
f	–	t	f	t	–	0.00402414	0.0	0.40120724	0.0

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	P(SecBreach = true)	P(SuspPeople = true)	P(Alarm = true)	P(Noise = true)
f	–	*t*	*f*	*t*	*t*	1.0	0.0	0.7	0.0
f	–	*t*	*f*	*t*	*f*	0.0	0.0	0.4	0.0
f	–	*t*	*f*	*f*	–	0.00855518	0.0	0.40256655	0.0
f	–	*t*	*f*	*f*	*t*	1.0	0.0	0.7	0.0
f	–	*t*	*f*	*f*	*f*	0.0	0.0	0.4	0.0
f	–	*f*	–	–	–	0.01	0.0	0.0198	0.0536425
f	–	*f*	–	–	*t*	1.0	0.0	0.99	0.1915
f	–	*f*	–	–	*f*	0.0	0.0	0.01	0.05225
f	–	*f*	–	*t*	–	0.01	0.0	0.0198	0.9503
f	–	*f*	–	*t*	*t*	1.0	0.0	0.99	0.98
f	–	*f*	–	*t*	*f*	0.0	0.0	0.01	0.95
f	–	*f*	–	*f*	–	0.01	0.0	0.0198	0.00645
f	–	*f*	–	*f*	*t*	1.0	0.0	0.99	0.15
f	–	*f*	–	*f*	*f*	0.0	0.0	0.01	0.005
f	–	*f*	*t*	–	–	0.03569931	0.0	0.04498532	1.0
f	–	*f*	*t*	–	*t*	1.0	0.0	0.99	1.0
f	–	*f*	*t*	–	*f*	0.0	0.0	0.01	1.0
f	–	*f*	*t*	*t*	–	0.01031253	0.0	0.02010628	1.0
f	–	*f*	*t*	*t*	*t*	1.0	0.0	0.99	1.0
f	–	*f*	*t*	*t*	*f*	0.0	0.0	0.01	1.0
f	–	*f*	*t*	*f*	–	0.23255814	0.0	0.23790698	1.0
f	–	*f*	*t*	*f*	*t*	1.0	0.0	0.99	1.0
f	–	*f*	*t*	*f*	*f*	0.0	0.0	0.01	1.0
f	–	*f*	*f*	–	–	0.00854328	0.0	0.01837242	0.0
f	–	*f*	*f*	–	*t*	1.0	0.0	0.99	0.0
f	–	*f*	*f*	–	*f*	0.0	0.0	0.01	0.0
f	–	*f*	*f*	*t*	–	0.00402414	0.0	0.01394366	0.0
f	–	*f*	*f*	*t*	*t*	1.0	0.0	0.99	0.0
f	–	*f*	*f*	*t*	*f*	0.0	0.0	0.01	0.0
f	–	*f*	*f*	*f*	–	0.00855518	0.0	0.01838408	0.0
f	–	*f*	*f*	*f*	*t*	1.0	0.0	0.99	0.0
f	–	*f*	*f*	*f*	*f*	0.0	0.0	0.01	0.0
f	*t*	–	–	–	–	0.25038501	0.0	1.0	0.08711611
f	*t*	–	–	–	*t*	1.0	0.0	1.0	0.1915
f	*t*	–	–	–	*f*	0.0	0.0	1.0	0.05225
f	*t*	–	–	*t*	–	0.25038501	0.0	1.0	0.95751155
f	*t*	–	–	*t*	*t*	1.0	0.0	1.0	0.98
f	*t*	–	–	*t*	*f*	0.0	0.0	1.0	0.95
f	*t*	–	–	*f*	–	0.25038501	0.0	1.0	0.04130583
f	*t*	–	–	*f*	*t*	1.0	0.0	1.0	0.15
f	*t*	–	–	*f*	*f*	0.0	0.0	1.0	0.005
f	*t*	–	*t*	–	–	0.55040024	0.0	1.0	1.0
f	*t*	–	*t*	–	*t*	1.0	0.0	1.0	1.0
f	*t*	–	*t*	–	*f*	0.0	0.0	1.0	1.0
f	*t*	–	*t*	*t*	–	0.25626564	0.0	1.0	1.0
f	*t*	–	*t*	*t*	*t*	1.0	0.0	1.0	1.0
f	*t*	–	*t*	*t*	*f*	0.0	0.0	1.0	1.0
f	*t*	–	*t*	*f*	–	0.90926038	0.0	1.0	1.0
f	*t*	–	*t*	*f*	*t*	1.0	0.0	1.0	1.0
f	*t*	–	*t*	*f*	*f*	0.0	0.0	1.0	1.0
f	*t*	–	*f*	–	–	0.22175469	0.0	1.0	0.0
f	*t*	–	*f*	–	*t*	1.0	0.0	1.0	0.0
f	*t*	–	*f*	–	*f*	0.0	0.0	1.0	0.0
f	*t*	–	*f*	*t*	–	0.11786027	0.0	1.0	0.0
f	*t*	–	*f*	*t*	*t*	1.0	0.0	1.0	0.0
f	*t*	–	*f*	*t*	*f*	0.0	0.0	1.0	0.0
f	*t*	–	*f*	*f*	–	0.22199703	0.0	1.0	0.0
f	*t*	–	*f*	*f*	*t*	1.0	0.0	1.0	0.0
f	*t*	–	*f*	*f*	*f*	0.0	0.0	1.0	0.0
f	*t*	*t*	–	–	–	0.01736973	0.0	1.0	0.05466873
f	*t*	*t*	–	–	*t*	1.0	0.0	1.0	0.1915
f	*t*	*t*	–	–	*f*	0.0	0.0	1.0	0.05225
f	*t*	*t*	–	*t*	–	0.01736973	0.0	1.0	0.95052109
f	*t*	*t*	–	*t*	*t*	1.0	0.0	1.0	0.98
f	*t*	*t*	–	*t*	*f*	0.0	0.0	1.0	0.95
f	*t*	*t*	–	*f*	–	0.01736973	0.0	1.0	0.00751861
f	*t*	*t*	–	*f*	*t*	1.0	0.0	1.0	0.15
f	*t*	*t*	–	*f*	*f*	0.0	0.0	1.0	0.005
f	*t*	*t*	*t*	–	–	0.0608447	0.0	1.0	1.0
f	*t*	*t*	*t*	–	*t*	1.0	0.0	1.0	1.0
f	*t*	*t*	*t*	–	*f*	0.0	0.0	1.0	1.0

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
f	t	t	t	t	–	0.01790842	0.0	1.0	1.0
f	t	t	t	t	t	1.0	0.0	1.0	1.0
f	t	t	t	t	f	0.0	0.0	1.0	1.0
f	t	t	t	f	–	0.34653465	0.0	1.0	1.0
f	t	t	t	f	t	1.0	0.0	1.0	1.0
f	t	t	t	f	f	0.0	0.0	1.0	1.0
f	t	t	f	–	–	0.01485556	0.0	1.0	0.0
f	t	t	f	–	t	1.0	0.0	1.0	0.0
f	t	t	f	–	f	0.0	0.0	1.0	0.0
f	t	t	f	t	–	0.00702106	0.0	1.0	0.0
f	t	t	f	t	t	1.0	0.0	1.0	0.0
f	t	t	f	t	f	0.0	0.0	1.0	0.0
f	t	t	f	f	–	0.01487612	0.0	1.0	0.0
f	t	t	f	f	t	1.0	0.0	1.0	0.0
f	t	t	f	f	f	0.0	0.0	1.0	0.0
f	t	f	–	–	–	0.5	0.0	1.0	0.121875
f	t	f	–	–	t	1.0	0.0	1.0	0.1915
f	t	f	–	–	f	0.0	0.0	1.0	0.05225
f	t	f	–	t	–	0.5	0.0	1.0	0.965
f	t	f	–	t	t	1.0	0.0	1.0	0.98
f	t	f	–	t	f	0.0	0.0	1.0	0.95
f	t	f	–	f	–	0.5	0.0	1.0	0.0775
f	t	f	–	f	t	1.0	0.0	1.0	0.15
f	t	f	–	f	f	0.0	0.0	1.0	0.005
f	t	f	t	–	–	0.78564103	0.0	1.0	1.0
f	t	f	t	–	t	1.0	0.0	1.0	1.0
f	t	f	t	–	f	0.0	0.0	1.0	1.0
f	t	f	t	t	–	0.50777202	0.0	1.0	1.0
f	t	f	t	t	t	1.0	0.0	1.0	1.0
f	t	f	t	t	f	0.0	0.0	1.0	1.0
f	t	f	t	f	–	0.96774194	0.0	1.0	1.0
f	t	f	t	f	t	1.0	0.0	1.0	1.0
f	t	f	t	f	f	0.0	0.0	1.0	1.0
f	t	f	f	–	–	0.46035587	0.0	1.0	0.0
f	t	f	f	–	t	1.0	0.0	1.0	0.0
f	t	f	f	–	f	0.0	0.0	1.0	0.0
f	t	f	f	t	–	0.28571429	0.0	1.0	0.0
f	t	f	f	t	t	1.0	0.0	1.0	0.0
f	t	f	f	t	f	0.0	0.0	1.0	0.0
f	t	f	f	f	–	0.46070461	0.0	1.0	0.0
f	t	f	f	f	t	1.0	0.0	1.0	0.0
f	t	f	f	f	f	0.0	0.0	1.0	0.0
f	f	–	–	–	–	0.00025493	0.0	0.0	0.0522855
f	f	–	–	–	t	1.0	0.0	0.0	0.1915
f	f	–	–	–	f	0.0	0.0	0.0	0.05225
f	f	–	–	t	–	0.00025493	0.0	0.0	0.95000765
f	f	–	–	t	t	1.0	0.0	0.0	0.98
f	f	–	–	t	f	0.0	0.0	0.0	0.95
f	f	–	–	f	–	0.00025493	0.0	0.0	0.00503697
f	f	–	–	f	t	1.0	0.0	0.0	0.15
f	f	–	–	f	f	0.0	0.0	0.0	0.005
f	f	–	t	–	–	0.00093371	0.0	0.0	1.0
f	f	–	t	–	t	1.0	0.0	0.0	1.0
f	f	–	t	–	f	0.0	0.0	0.0	1.0
f	f	–	t	t	–	0.00026298	0.0	0.0	1.0
f	f	–	t	t	t	1.0	0.0	0.0	1.0
f	f	–	t	t	f	0.0	0.0	0.0	1.0
f	f	–	t	f	–	0.00759184	0.0	0.0	1.0
f	f	–	t	f	t	1.0	0.0	0.0	1.0
f	f	–	t	f	f	0.0	0.0	0.0	1.0
f	f	–	f	–	–	0.00021748	0.0	0.0	0.0
f	f	–	f	–	t	1.0	0.0	0.0	0.0
f	f	–	f	–	f	0.0	0.0	0.0	0.0
f	f	–	f	t	–	0.00010199	0.0	0.0	0.0
f	f	–	f	t	t	1.0	0.0	0.0	0.0
f	f	–	f	t	f	0.0	0.0	0.0	0.0
f	f	–	f	f	–	0.00021779	0.0	0.0	0.0
f	f	–	f	f	t	1.0	0.0	0.0	0.0
f	f	–	f	f	f	0.0	0.0	0.0	0.0
f	f	t	–	–	–	0.00502513	0.0	0.0	0.05294975
f	f	t	–	–	t	1.0	0.0	0.0	0.1915

Continued on next page

Table A.1 – *Continued from previous page*

SuspPeople	*Alarm*	*ElecMalfct*	*Noise*	*Accident*	*SecBreach*	*P(SecBreach = true)*	*P(SuspPeople = true)*	*P(Alarm = true)*	*P(Noise = true)*
f	*f*	*t*	–	–	*f*	0.0	0.0	0.0	0.05225
f	*f*	*t*	–	*t*	–	0.00502513	0.0	0.0	0.95015075
f	*f*	*t*	–	*t*	*t*	1.0	0.0	0.0	0.98
f	*f*	*t*	–	*t*	*f*	0.0	0.0	0.0	0.95
f	*f*	*t*	–	*f*	–	0.00502513	0.0	0.0	0.00572864
f	*f*	*t*	–	*f*	*t*	1.0	0.0	0.0	0.15
f	*f*	*t*	–	*f*	*f*	0.0	0.0	0.0	0.005
f	*f*	*t*	*t*	–	–	0.01817405	0.0	0.0	1.0
f	*f*	*t*	*t*	–	*t*	1.0	0.0	0.0	1.0
f	*f*	*t*	*t*	–	*f*	0.0	0.0	0.0	1.0
f	*f*	*t*	*t*	*t*	–	0.00518299	0.0	0.0	1.0
f	*f*	*t*	*t*	*t*	*t*	1.0	0.0	0.0	1.0
f	*f*	*t*	*t*	*t*	*f*	0.0	0.0	0.0	1.0
f	*f*	*t*	*t*	*f*	–	0.13157895	0.0	0.0	1.0
f	*f*	*t*	*t*	*f*	*t*	1.0	0.0	0.0	1.0
f	*f*	*t*	*t*	*f*	*f*	0.0	0.0	0.0	1.0
f	*f*	*t*	*f*	–	–	0.00428997	0.0	0.0	0.0
f	*f*	*t*	*f*	–	*t*	1.0	0.0	0.0	0.0
f	*f*	*t*	*f*	–	*f*	0.0	0.0	0.0	0.0
f	*f*	*t*	*f*	*t*	–	0.00201613	0.0	0.0	0.0
f	*f*	*t*	*f*	*t*	*t*	1.0	0.0	0.0	0.0
f	*f*	*t*	*f*	*t*	*f*	0.0	0.0	0.0	0.0
f	*f*	*t*	*f*	*f*	–	0.00429597	0.0	0.0	0.0
f	*f*	*t*	*f*	*f*	*t*	1.0	0.0	0.0	0.0
f	*f*	*t*	*f*	*f*	*f*	0.0	0.0	0.0	0.0
f	*f*	*f*	–	–	–	0.00010202	0.0	0.0	0.05226421
f	*f*	*f*	–	–	*t*	1.0	0.0	0.0	0.1915
f	*f*	*f*	–	–	*f*	0.0	0.0	0.0	0.05225
f	*f*	*f*	–	*t*	–	0.00010202	0.0	0.0	0.95000306
f	*f*	*f*	–	*t*	*t*	1.0	0.0	0.0	0.98
f	*f*	*f*	–	*t*	*f*	0.0	0.0	0.0	0.95
f	*f*	*f*	–	*f*	–	0.00010202	0.0	0.0	0.00501479
f	*f*	*f*	–	*f*	*t*	1.0	0.0	0.0	0.15
f	*f*	*f*	–	*f*	*f*	0.0	0.0	0.0	0.005
f	*f*	*f*	*t*	–	–	0.00037381	0.0	0.0	1.0
f	*f*	*f*	*t*	–	*t*	1.0	0.0	0.0	1.0
f	*f*	*f*	*t*	–	*f*	0.0	0.0	0.0	1.0
f	*f*	*f*	*t*	*t*	–	0.00010524	0.0	0.0	1.0
f	*f*	*f*	*t*	*t*	*t*	1.0	0.0	0.0	1.0
f	*f*	*f*	*t*	*t*	*f*	0.0	0.0	0.0	1.0
f	*f*	*f*	*t*	*f*	–	0.00305157	0.0	0.0	1.0
f	*f*	*f*	*t*	*f*	*t*	1.0	0.0	0.0	1.0
f	*f*	*f*	*t*	*f*	*f*	0.0	0.0	0.0	1.0
f	*f*	*f*	*f*	–	–	0.00008703	0.0	0.0	0.0
f	*f*	*f*	*f*	–	*t*	1.0	0.0	0.0	0.0
f	*f*	*f*	*f*	–	*f*	0.0	0.0	0.0	0.0
f	*f*	*f*	*f*	*t*	–	0.00004081	0.0	0.0	0.0
f	*f*	*f*	*f*	*t*	*t*	1.0	0.0	0.0	0.0
f	*f*	*f*	*f*	*t*	*f*	0.0	0.0	0.0	0.0
f	*f*	*f*	*f*	*f*	–	0.00008715	0.0	0.0	0.0
f	*f*	*f*	*f*	*f*	*t*	1.0	0.0	0.0	0.0
f	*f*	*f*	*f*	*f*	*f*	0.0	0.0	0.0	0.0

Table A.1: Probabilities for $SecurityBreach = true$, $SuspiciousPeople = true$, $Alarm = true$ and $Noise = true$ under each of the possible evidence scenarios for the Bayesian Network representing the *Watchman* agent's beliefs

REFERENCES

BLUM, A. L.; FURST, M. L. Fast Planning Through Planning Graph Analysis. **Artificial Intelligence**, [S.l.], v.90, p.281–300, 2 1997.

BLUM, A.; LANGFORD, J. C. Probabilistic Planning in the Graphplan Framework. In: FIFTH EUROPEAN CONFERENCE ON PLANNING. **Proceedings...** [S.l.: s.n.], 1998.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak Using Jason**. [S.l.]: John Wiley & Sons, Ltd, 2007.

BRATMAN, M.; ISRAEL, D. J.; POLLACK, M. E. Plans and resource-bounded practical reasoning. **Computational Intelligence**, [S.l.], v.4, p.349–355, 1988.

CARRERA, Á.; IGLESIAS, C. A. B2DI A Bayesian BDI Agent Model with Causal Belief Updating based on MSBN. In: INTERNATIONAL CONFERENCE ON AGENTS AND ARTIFICIAL INTELLIGENCE (ICAART2012), 4. **Proceedings...** SciTePress, 2012. p.343–346.

COZMAN, F. G. Generalizing Variable Elimination in Bayesian Networks. **Workshop on Probabilistic Reasoning in Artificial Intelligence**, [S.l.], 2000.

FAGUNDES, M. S.; VICARI, R. M. **Integrating BDI Model and Bayesian Networks**. 2007. Dissertação (Mestrado em Ciência da Computação) — UFRGS.

FAGUNDES, M. S.; VICARI, R. M.; COELHO, H. Deliberation Process in a BDI Model with Bayesian Networks. In: PRIMA. **Anais...** Springer, 2007. p.207–218. (Lecture Notes in Computer Science, v.5044).

JENNINGS, N. R. On agent-based software engineering. **Artificial Intelligence**, [S.l.], v.117, p.277–296, 2000.

JENSEN, F. V.; NIELSEN, T. D. **Bayesian Networks and Decision Graphs**. 2nd.ed. [S.l.]: Springer, 2007.

KIELING, G.; VICARI, R. M. **Inserção de Conhecimento Probabilístico para Construção de Agentes BDI Modelados em Redes Bayesianas**. 2010. Dissertação (Mestrado em Ciência da Computação) — UFRGS.

KIELING, G.; VICARI, R. M. Insertion of Probabilistic Knowledge into BDI Agents Construction Modeled in Bayesian Networks. In: INTERNATIONAL CONFERENCE ON COMPLEX, INTELLIGENT, AND SOFTWARE INTENSIVE SYSTEMS (CISIS 2011), Seoul. **Anais...** California: Conference Publishing Services (CPS), 2011. v.1, p.115–122.

KINNY, D. N.; GEORGEFF, M. P. Commitment and Effectiveness of Situated Agents. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 12., Sydney, Australia. **Proceedings...** [S.l.: s.n.], 1991. p.82–88.

LITTLE, I.; THIÉBAUX, S. Concurrent Probabilistic Planning in the Graphplan Framework. **16th International Conference on Automated Planning and Scheduling**, [S.l.], 6 2006.

MACHADO, G. M.; VICARI, R. M. **JAmplia - Uma proposta de ambiente multiagente probabilístico inteligente em Java voltado para WEB**. 2006.

MENEGUZZI, F. R. **Extending agent languages for multiagent domains**. 2009. Tese (Doutorado em Ciência da Computação) — King's College London.

MENEGUZZI, F. R. et al. Incorporating Planning into BDI Agents. **Scalable Computing: Practice and Experience**, [S.l.], v.8, p.15–28, 2007.

MÓRA, M. C. et al. BDI Models and Systems: reducing the gap. In: INTELLIGENT AGENTS V: AGENTS THEORIES, ARCHITECTURES, AND LANGUAGES, 5TH INTERNATIONAL WORKSHOP, ATAL '98. **Anais...** Springer Berlin Heidelberg, 1999. p.11–27.

NEBEL, B. On the Compilability and Expressive Power of Propositional Planning Formalisms. **Journal of Artificial Intelligence Research**, [S.l.], v.12, p.271–315, 2000.

PEARL, J. **Probabilistic Reasoning in Intelligent Systems**: networks of plausible inference. San Mateo, CA: Morgan-Kaufmann, 1988.

PITTSBURGH, D. S. L. U. of. **GeNIe & SMILE.** Disponível em: < http://genie.sis.pitt.edu/ >. Acesso em: 15 jul. 2013.

RAO, A. S. AgentSpeak(L): BDI agents speak out in a logical computable language. In: VELDE, W. V. de; PERRAM, J. W. (Ed.). **Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World**. [S.l.]: Springer-Verlag, 1996. p.42–55. (LNCS, v.1038).

RAO, A. S.; GEORGEFF, M. P. BDI Agents: from theory to practice. In: FIRST INTERNATIONAL CONFERENCE ON MULTIAGENT SYSTEMS. **Proceedings...** [S.l.: s.n.], 1995.

RUSSEL, S. J.; NORVIG, P. **Artificial Intelligence**: a modern approach. New Jersey: Prentice Hall, 1994.