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

MOSER SILVA FAGUNDES

Integrating BDI Model and Bayesian Networks

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

Prof. Dra. Rosa Maria Vicari
Advisor

Porto Alegre, March 2007

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Fagundes, Moser Silva

Integrating BDI Model and Bayesian Networks / Moser Silva Fagundes. – Porto Alegre: PPGC da UFRGS, 2007.

104 f.: il.

Thesis(Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Advisor: Rosa Maria Vicari.

1. BDI Model. 2. Bayesian Networks. 3. Ontologies. I. Vicari, Rosa Maria. II. Título.

“It is the mark of an educated mind to be able to entertain a thought without accepting it.”

— Aristotle

ACKNOWLEDGEMENTS

First, thanks to my advisor Rosa Vicari for your friendship, kindness, professionalism and patience. Without your support, advices and criticisms this dissertation would never happen.

Thanks to my colleague and friend Elder Santos for helping to shape my ideas about this research. The long talks about philosophic and sport issues were of important value to keep my mind sane in those stressing periods before the innumerous deadlines.

Thanks to my colleagues and friends Evandro Miletto, Luís Carlos Fonseca, Elisa Boff and Edilson Pontarolo. Your friendship along these two years has been priceless.

Thanks my friends Lorenzo and Rafael. I am completely sure that I will remember this time over my entire life. A lot of histories to tell our grandsons…

Thank you my grandparents: Epitácio, Neli, Silvio (in memoriam) and Odila.

Last, but not at least, I thank my parents Carmiranda and José. Thanks for supporting my decisions, celebrating my victories and bringing me relief in those bad moments. Words could never express what I feel for you.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

ACL	Agent Communication Language
AI	Artificial Intelligence
AIG	Artificial Intelligence Group
API	Application Programming Interface
BDI	Belief-Desire-Intention
CORBA	Common Object Request Broker Architecture
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
DAML	DARPA Agent Mark-up Language
DARPA	Defense Advanced Research Projects Agency
DL	Description Logic
FIPA	Foundation for Intelligent Physical Agents
HTML	Hypertext Markup Language
FIPA-ACL	FIPA - Agent Communication Language
KQML	Knowledge Query Manipulation Language
OIL	Ontology Inference Layer
OWL	Web Ontology Language
PACL	Probabilistic Agent Communication Language
PRS	Procedural Reasoning System
PR-OWL	Probabilistic-OWL
RDF	Resource Description Framework
RDFS	RDF Schema
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Individually, Artificial Intelligence research areas have proposed approaches to solve several complex real-world problems. The agent-based paradigm provided autonomous agents, capable of perceiving their environment, reacting in accordance with different situations, and establishing social interactions with other software agents and humans. Bayesian networks provided a way to represent graphically the conditional probability distributions and an evidence-based probabilistic reasoning. Ontologies are an effort to develop formal and explicit specifications of concepts, which have been used by a wide range of research areas, including Multiagent Systems.

However, there are applications whose requirements can not be addressed by a single technology. Circumstances like these demand the integration of technologies developed by distinct areas of Computer Science. This work is particularly concerned with the integration of Belief-Desire-Intention (BDI) agent architecture and Bayesian networks. Moreover, it is adopted an ontology-based approach to represent the agent's uncertain knowledge.

To bring together those technologies, it was developed an ontology to represent the structure of Bayesian networks knowledge representation. This ontology supports the interoperability among agents that comply with the proposed architecture, and it also facilitates the understanding necessary to abstract the agents' mental states and cognitive processes through elements of Bayesian networks. Once specified the ontology, it was integrated with the BDI agent architecture. By integrating BDI architecture and Bayesian networks, it was obtained a cognitive agent architecture capable of reasoning under uncertainty. It was performed in two stages: abstraction of mental states through Bayesian networks and specification of the deliberative process.

Finally, it was developed a case study, which consists in applying the probabilistic BDI architecture in the Social Agent, a component of a multiagent educational portal (PortEdu).

Keywords: BDI Model, Bayesian Networks, Ontologies.

Integrando Modelo BDI e Redes Bayesianas

RESUMO

Individualmente, as linhas de pesquisa da Inteligência Artificial têm proposto abordagens para a resolução de inúmeros problemas complexos do mundo real. O paradigma orientado a agentes provê os agentes autônomos, capazes de perceber os seus ambientes, reagir de acordo com diferentes circunstâncias e estabelecer interações sociais com outros agentes de software ou humanos. As redes Bayesianas fornecem uma maneira de representar graficamente as distribuições de probabilidades condicionais e permitem a realização de raciocínios probabilísticos baseados em evidências. As ontologias são especificações explícitas e formais de conceituações, que são usadas em uma variedade de áreas de pesquisa, incluindo os Sistemas Multiagentes.

Contudo, existem aplicações cujos requisitos não podem ser atendidos por uma única tecnologia. Circunstâncias como estas exigem a integração de tecnologias desenvolvidas por distintas áreas da Ciência da Computação. Esta dissertação trata a integração do modelo de agentes BDI (*Belief-Desire-Intention*) e das redes Bayesianas. Além disso, é adotada uma abordagem baseada em ontologias para representar o conhecimento incerto dos agentes.

O primeiro passo em direção a integração foi o desenvolvimento de uma ontologia para representar a estrutura das redes Bayesinas. Esta ontologia tem como principal objetivo permitir a interoperabilidade agentes compatíveis com a arquitetura proposta. No entanto, a ontologia também facilita o entendimento necessário para abstrair os estados mentais e processos cognitivos dos agentes através de elementos das redes Bayesianas. Uma vez construída a ontologia, a mesma foi integrada com a arquitetura BDI. Através da integração do modelo BDI com as redes Bayesianas foi obtida uma arquitetura cognitiva de agentes capaz de deliberar sob incerteza. O processo de integração foi composto de duas etapas: abstração dos estados mentais através de elementos das redes Bayesianas e especificação do processo deliberativo.

Finalmente, foi desenvolvido um estudo de caso, que consistiu na aplicação da arquitetura proposta no Agente Social, um componente de um portal educacional multiagente (PortEdu).

Palavras-chave: Modelo BDI, Redes Bayesianas, Ontologias.

1 INTRODUCTION

The present master dissertation is contextualized in the Artificial Intelligence (AI), more specifically, in the Autonomous Agents and Multiagent Systems area. However, it also involves researches on ontologies and Bayesian networks. Each mentioned area addresses specific issues, in example: the agent-oriented paradigm provides autonomous agents, capable of perceiving their environment, reacting in accordance with the situation, and establishing social interactions with other software agents and humans; Bayesian networks correspond to graphical representations of conditional probability distributions, which enable an evidence-based probabilistic reasoning; and ontologies are an effort to specify concepts, which can be shared between applications in order to achieve an interoperability at semantic level.

Individually, those knowledge areas have proposed solutions to several complex real-world problems. However, there are circumstances that require features exhibited by distinct technologies. Circumstances like these demand the integration of distinct knowledge areas to fulfill the application requirements. This research is particularly concerned with the integration of the Belief-Desire-Intention (BDI) agent architecture and the Bayesian networks. Moreover, it adopts an ontology-based approach to represent the agent's uncertain knowledge. Throughout this dissertation, it is discussed and highlighted the synergistic effects of integrating those technologies.

Ontologies have been used by Computer Science to deal with several tasks, such as conceptual modeling and interoperability promotion. Currently, the ontology research is closely related to the Semantic Web (BERNERS-LEE; HENDLER; LASSILA, 2001). The purpose of the Semantic Web is to aggregate meaning to Web pages, in a way that not only humans, but also computer software may interpret its content. Considering the Semantic Web as an open system, populated by autonomous agents carrying out activities in behalf of its owners, interoperability issues (i.e. how these autonomous agents from distinct domains and with distinct goals will share their knowledge, cooperate and maximize the utility of the whole system) arise.

Historically, traditional knowledge representation formalisms did not take into account uncertainty. Examples of those languages include the W3C (World Wide Web Consortium) standard for Semantic Web, the OWL (Web Ontology Language), and its predecessors XML (eXtensible Markup Language) and RDF (Resource Description Framework). Efforts to represent probabilistic information within ontologies (i.e. PR-OWL (Probabilistic-OWL) (COSTA; LASKEY, 2006)) have been performed in the Semantic Web context. However, this lack of support is also noticed in other areas, such as BDI agent systems.

Most proposals of BDI agent architectures do not concern with the representation of the uncertainty inherent to the agent's environment (GEORGEFF; INGRAND, 1989;

JENNINGS et al., 1992; MÜLLER, 1996; RAO, 1996), despite the fact that real-world applications have to deal often with uncertain and imprecise information. Resource-bounded agents that inhabit complex and dynamic environments (i.e. Internet) can not always have accurate information about it. Consequently, to take a rational action in a fashion time becomes a challenge.

1.1 Related Research

The work presented in (SANTOS, 2006; SANTOS; BOFF; VICARI, 2006) uses ontologies to promote the semantic interoperability among heterogeneous agents. It defines an ontology that covers elementary aspects necessary to construct Bayesian network individuals, but it does not specify the complete structure of that knowledge representation. By this it means that the network's elements were partially specified. Moreover, this work defines an engine to automatically generate OWL individuals from Bayesian networks constructed with the HUGIN (JENSEN et al., 2002) tool. This conversion mechanism, together with the ontology, allows the interoperability with other Bayesian agents through the OWL standard. Summarizing, this work focuses on providing a framework to develop interoperable Bayesian agents.

Another work developed in the scope of the Artificial Intelligence Group (AIG) of the Universidade Federal do Rio Grande do Sul (UFRGS) is the Probabilistic Agent Communication Language (PACL) (GLUZ et al., 2006). It is an extension of the FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Language) (FIPA: Foundation For Intelligent Physical Agents, 2002) designed to deal with probabilistic knowledge communication. The PACL specifies new axioms that are necessary to accomplish the probabilistic communication. Besides the axioms, the language also designs assertive and directive probabilistic speech acts, which extends FIPA-ACL. The PACL provides a way to communicate probabilistic knowledge by extending the FIPA-ACL, allowing more expressiveness to this language. It does not deal with the communication of uncertainty at the message content level, concerning how different Bayesian agents might exchange knowledge regarding their networks and evidences.

BayesOWL (PAN et al., 2005) was developed to handle the issue of automatic ontology mapping. This approach defines additional markups that can add probabilities to concepts, individuals, properties and its relationships. It also defines a set of translation rules to convert the probabilistic annotated ontology into a Bayesian network. The focus on ontology mapping limits the BayesOWL markups since it was not necessary to represent variables with states different than true or false. The reason for this is that the probabilistic knowledge associated with each ontology concept was used only for telling if two concepts from different ontologies were the same.

Another approach that represents probabilistic knowledge through OWL is PR-OWL (COSTA; LASKEY, 2006). It aims to provide a framework for probabilistic ontologies. It constitutes an extension of OWL to express probabilistic knowledge. The PR-OWL language adds new definitions to OWL allowing the expression of uncertainty. The need for standardization represents a drawback for short-term solutions but also points to a very interesting medium to long term solution, as it fits well (providing the formal foundation of a first-order logic) in the W3C model of standards.

The Petri net ontology (GASEVIC; DEVEDZIC, 2006) has common aspects with the Bayesian network ontology proposed by this dissertation, since there are similarities in the conceptual modeling task. The objective of that research is to provide the necessary structure to share Petri nets on the Semantic Web context. This work reviews previous efforts done in Petri net sharing and Petri net formalizations. Then, it specifies a Petri net ontology using OWL language. Another work concerning Petri net representation is (BRETON; BÉZIVIN, 2001). Its main goal is the understanding of the model executability. In order to achieve this goal, it discusses Petri net related concepts, classifying them in static or dynamic. The final result is a three level Petri net metamodel. The first level is the definition metamodel that specifies the static part of the nets. The second level defines a particular situation of a Petri net. The third level is an execution metamodel that defines a sequence of situations.

The AgentSpeak(L) (RAO, 1996) is a language that can be viewed as an abstraction of the PRS (Procedural Reasoning System) (GEORGEFF; INGRAND, 1989). This agent model represents beliefs through first-order logic. The plans are composed by a trigger, a context and a body. The trigger can be the insertion or removal of a belief or an object. The context specifies under which circumstances a plan can be applied. The body contains a sequence of actions.

BayesJason (CALCIN, 2006) is an extension of the Jason (BORDINI et al., 2005) tool that addresses the integration of Bayesian networks and BDI agents. Jason is a java-based AgentSpeak(L) interpreter that allows the development of BDI agents and their execution over the Internet as a distributed system. The BayesJason implements a probabilistic module for Jason, enabling BDI agents to act on uncertain environments. To allow the programming of Bayesian networks within the Jason, the AgentSpeak(L) grammar was modified and extended. In the new grammar, beliefs have probabilities and each belief corresponds to a Bayesian network node. The Jason's API (Application Programming Interface) was extended to allow the dynamic specification of Bayesian networks, the development of functions to handle the perceived evidences, and the execution of probabilistic inference on the networks. Finally, it was implemented new internal actions to manipulate the networks. Although integrating Bayesian networks and BDI model, the BayesJason does not explore the features of the Bayesian networks to improve the cognitive processes. The networks are used to associate probabilities with beliefs and to select the best predefined plan according to the circumstance.

1.2 Motivation

This dissertation is developed in the context of the Artificial Intelligence Group (AIG) of the Universidade Federal do Rio Grande do Sul (UFRGS), supervised by the professor Rosa Vicari. The AIG have been researching and constructing educational systems, using AI techniques to propose solutions to issues that have not been addressed by other Computer Science areas.

The first step of this work was motivated by the ontology-based approach proposed by Santos (SANTOS, 2006) to enhance the interoperability of the PortEdu agents. One of the pointed future works consists in remodeling and extending the ontology to represent the complete structure of a Bayesian network. By doing this, the abstraction of mental states through Bayesian networks was clear, since the ontology allowed the unambiguously understanding of that probabilistic model.

By performing a case study, which consisted of applying the ontology in the Social Agent (SANTOS; FAGUNDES; VICARI, 2007), it was detected a lack between the knowledge representation and the goal-oriented behavior. It was the motivation of integrating BDI architecture and Bayesian networks.

Throughout the probabilistic BDI architecture development, it was realized that most agents that represent knowledge through Bayesian networks have reasoning processes dependent of the network instances. It makes impossible the reuse of that component. Bayesian networks share a common structure (causal relations, conditional probability tables, chance variables, mutually exclusive states), which can be explored to specify a deliberation process.

Finally, the results obtained by this work contribute directly to the AIG researches. The integration of Bayesian networks and BDI agent architecture will be applied in the Social Agent, a component of PortEdu (NAKAYAMA; VICARI; COELHO, 2005). Besides, the approach can be applied to the remaining PortEdu agents.

1.3 Objectives

The general objective of this research is to enable the BDI agent architecture to operate, in an interoperable way, on environments where uncertain information are present. In order to attain the general objective, the following specific objectives were proposed:

- Development of an ontology that specifies the Bayesian network concepts to address the semantic interoperability among BDI agents.
- Representation of the BDI mental states through Bayesian networks, more specifically, though the Bayesian network ontology. This abstraction is the first step towards the probabilistic BDI model.
- Specification of the cognitive processes taking into account that the beliefs correspond to Bayesian networks, and the desires and intentions correspond to particular states of chance variables that agents intend to reach.
- Development of the Social Agent case study to demonstrate the integration of technologies proposed by this work.

1.4 Contribution

The most expressive contribution of this research is the probabilistic BDI model, which is capable of reasoning under uncertainty. Differently from BayesJason, which abstracts beliefs through Bayesian networks and proactive mental states through plans, this work abstracts beliefs, desires and intentions through Bayesian networks. The deliberative process described in this dissertation uses the knowledge representation to reason, instead of particular Bayesian network instances. This feature enables the reuse of this cognitive process.

It is possible to claim that the Bayesian network ontology is itself an independent contribution, since it can be employed in several applications beyond the architecture here presented. The understanding about the Bayesian networks' domain provided by the ontology, more specifically by conceptual modeling task, facilitated the integration of that knowledge representation with other technologies, such as the BDI model here presented.

1.5 Organization

This dissertation is organized as follows: the Chapter 2 presents the foundations that underlie this research; the Chapter 3 presents the specification of the Bayesian network ontology; the Chapter 4 presents the integration of the BDI model and the Bayesian networks; the Social Agent case study is detailed in the Chapter 5; finally, the Chapter 6 presents the conclusion and future work.

2 FOUNDATIONS

This Chapter presents the foundations that underlie this dissertation. The Section 2.1 presents the discrete Bayesian networks, which representation is modeled through an ontology described in the Chapter 3. To provide the theory necessary to understand the model described in the Chapter 3, the Section 2.2 presents the ontology theoretical references. Finally, considering that the Chapter 4 proposes a BDI architecture, the Section 2.3 complements the foundations by presenting the BDI model theory.

2.1 Discrete Bayesian Networks

For Jensen (JENSEN, 2001), a Bayesian network consists of the following:

- A set of variables and a set of directed arcs between variables.
- Each variable has a set of mutually exclusive states.
- The variables together with the directed arcs form a directed acyclic graph (DAG).
- To each variable A with parents B1,...,Bn, there is attached the potential table P(A|B1,...,Bn).

Discrete Bayesian networks can contain only *chance variables*, which represent an exhaustive set of random events, referred to as domain of the variable. In Bayesian networks, the notion of variables and nodes are often used interchangeably. For models that contain decision nodes and utility functions it is convenient to differentiate nodes and variables, since in those models a node does not necessarily represents a variable. States of chance variables are not necessarily Boolean. They can represent numerical values, intervals, or labels, beyond the Boolean values true and false.

Since prior chance variables do not have parent nodes, their probability table is reduced to unconditional probabilities. This kind of variable is only *conditioning*, since it is not *conditioned* by other variables. On the other hand, non prior nodes are always *conditioned* by their parent nodes. In the Figure 2.1, the variables *Earthquake* and *Burglary* are conditioning variables. The *RadioNews* variable is conditioned by the *Earthquake* variable. The *Alarm* variable is conditioned by *Burglary* and *Earthquake*, and *WatsonCalls* is conditioned by *Alarm*.

The Figure 2.1 illustrates the Bayesian network example *Burglary or Earthquake* (PEARL, 1988). In this very known example, Mr. Holmes is working in his office when he receives a phone call from his neighbor Dr. Watson, who tells Mr. Holmes that his alarm has gone off. Convinced that a burglar has broken into his house, Mr. Holmes rushes to his car and heads for home. On his way home, he listens to the radio, and in the news it is reported that there has been a small earthquake in the area. Knowing that earthquakes have a tendency to make burglar alarms go off, he returns to his work.

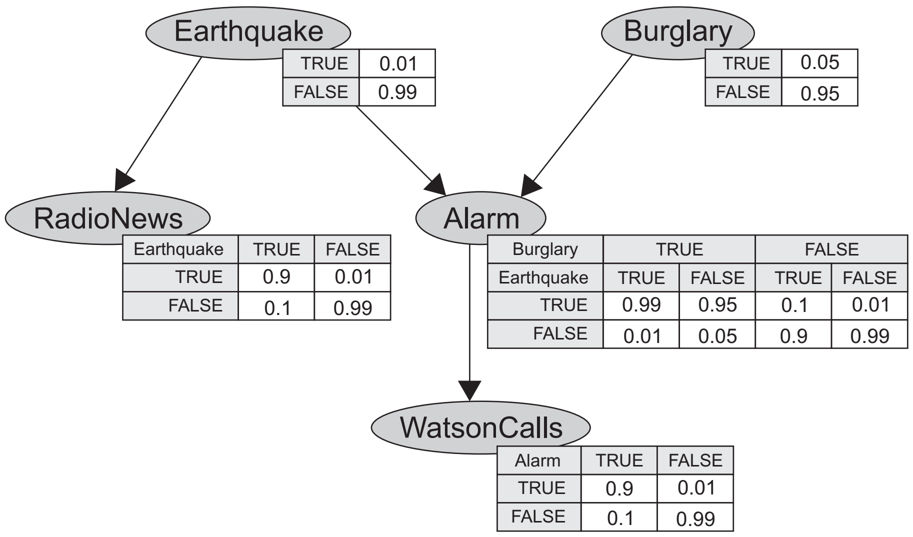


Figure 2.1: Burglary or Earthquake Bayesian network.

Bayesian networks are used for calculating chances when you perceive information about the states of the network's variables. These information are named *evidences*. They can be classified in *hard evidences* or *soft evidences*. Evidences that assign zero probability to all except one state of a chance variable are considered hard evidences; otherwise they are considered soft evidences. Hard evidences indicate that a particular state has been observed.

2.1.1 Connections and D-Separation Criterion

The Figure 2.2 depicts the three kinds of connection among nodes of Bayesian networks: *serial*, *converging* and *diverging*. In a *serial* connection, a hard evidence on the middle variable (*Alarm*) will block the flow of information. In other words, if the state of *Alarm* variable is known, an evidence on *Burglary* will not affect our beliefs about *WatsonCalls*. Otherwise, if there is not evidences on the middle variable (*Alarm*), an evidence on *Burglary* will affect our beliefs about *Alarm* and consequently about *WatsonCalls*.

In a *diverging* connection, a hard evidence on the *Earthquake* will block the flow of information, and the inexistence of evidences will allow the flow. Assuming that there is not evidence on *Earthquake*, it is possible to claim that an evidence on *Alarm* will affect our beliefs about *Earthquake*, since earthquake is an explanation for alarm. Once affected our beliefs about *Alarm*, our beliefs about *RadioNews* will be also affected.

Finally, the *converging* connection will allow the flow of information under the availability of evidences (possibly soft) in the middle variable, in that example, *Alarm*. It is possible to conclude that if a descendant of *Alarm* (i.e. *WatsonCalls*) receives an evidence, *Alarm* will be affected, and the flow of information through *Alarm* allowed. The flow of information in the converging connections is blocked if no evidences are available on the middle variable. This happens because it is not possible to infer the chances of a state of a parent node (*Earthquake*) given an evidence on another parent node (*Burglary*). On the other hand, if the *Alarm* goes off, information about *Burglary* will confirm or dismiss *Earthquake* as the cause of *Alarm*. This property of converging connections, that information about a state of *Earthquake* provides an explanation for an observed event on *Alarm*, and hence confirms or dismisses *Burglary*, is often referred to as the explaining away or intercausal inference.

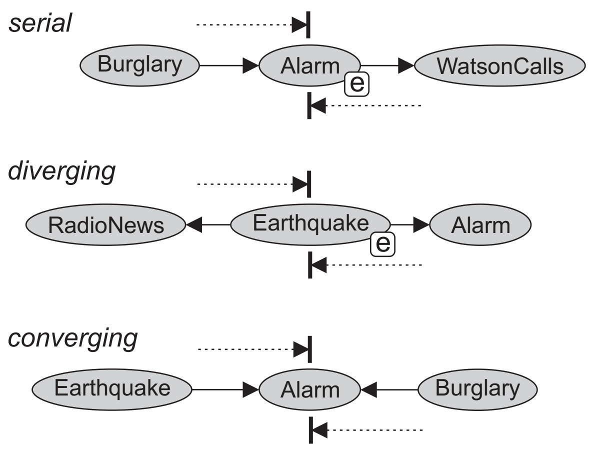


Figure 2.2: Serial, Diverging and Converging connections.

Besides intercausal inference, Bayesian networks allow causal reasoning (deductive) and diagnostic reasoning (abductive). The causal reasoning follows the direction of the arcs between variables. If it is observed that the *Alarm* has gone off, the beliefs about *WatsonCalls* are updated. The abductive reasoning goes against the direction of the causal arcs. An example of abductive reasoning consists in updating *Alarm* beliefs given an observation on *WatsonCalls*.

Once presented the three connections among variables, and consequently, the three possible cases in which evidence may be transmitted through a variable, it is presented the *d-separation* criterion. Jensen (JENSEN, 2001) asserts that two distinct variables A and B in a causal network are d-separated if, for all paths between A and B, there is an intermediate variable V (distinct from A and B) such that either

- the connection is serial or diverging and V has received evidence, or
- the connection is converging, and neither V nor any of V's descendents have received evidence.

2.1.2 Probabilities

Bayesian networks have *qualitative* and *quantitative* aspects. Qualitative aspects are defined by the graphical (topological) characteristics of the network. They relate to the causal relations among variables. The quantitative aspects are defined by probabilities. They are the numerical part, which relates to the uncertainty representation. Assuming that Bayesian networks can be expressed as joint probability distributions, it is introduced the probabilistic foundations that support this model.

The *conditional probability* concept is crucial to understand the Bayesian networks. It means that given an event *b*, the probability of occurrence of an event *a* is *x*. In example, the probability of *RadioNews* given *Earthquake* is *0.9*:

P(*RadioNews*=true| *Earthquake*=true) = *0.9*.

Each non prior node of a Bayesian network has a Conditional Probability Table (CPT). This table contains the probabilities of occurrence of a state (event) given the conditions. These conditions are imposed by the parent nodes. The Table 2.1 depicts the *Alarm* variable's CPT. In example, if the *Burglary* and *Earthquake* have been observed, then the probability of the *Alarm* going off is *0.99*.

Table 2.1: Conditional Probability Table for the Alarm variable.

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

To solve a Bayesian network is to compute all posterior marginal distributions given a set (possibly empty) of evidences. This process is known as Bayesian inference. The Figure 2.3 illustrates the computed marginal distributions for the network presented in the Figure 2.1.

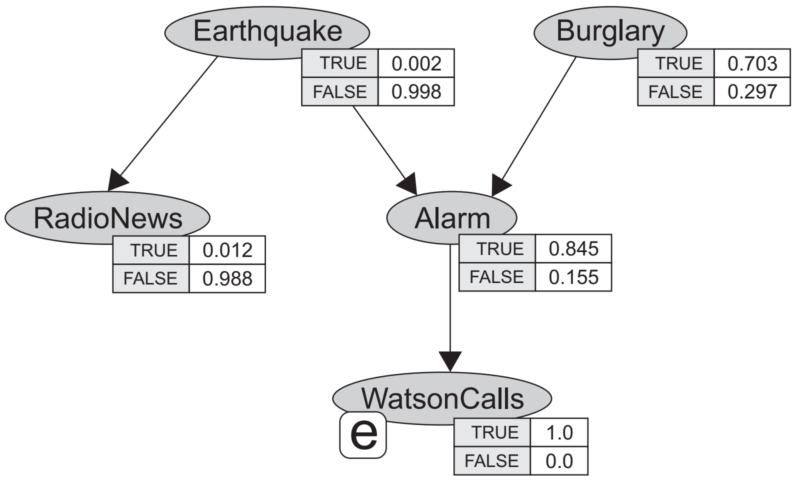


Figure 2.3: Computed marginal distributions for *Burglary or Earthquake* Bayesian network.

Instead of the conditional probability tables, it is presented the marginal distribution tables. The network depicted in the Figure 2.3 has an evidence indicating that the state *TRUE* of *WatsonCalls* has been observed. The inference process takes into account this evidence to estimate the chances of the remaining variables. For details about the inference process, see (COWELL et al., 1999; JENSEN, 2001).

2.2 Ontologies

Guarino (GUARINO, 1997) suggests the following ontology definition: an ontology is an explicit, partial account of a conceptualization.

Another definition of ontology is proposed by (SCHREIBER; WIELINGA; JANSWEIJER, 1995): an ontology is an explicit, partial specification of a conceptualization that is expressible as a meta-level viewpoint on a set of possible domain theories for the purpose of modular design, redesign and reuse of knowledge-intensive system components.

The ontology definition is closely related with the conceptualization definition. A conceptualization consists in defining concepts (entities, attributes, processes) and their inter-relationships (USCHOLD; GRUNINGER, 1996).

Ontologies, independently of representation language, share common features, including concepts and properties. A concept (class) is an abstract idea that denotes all of the objects in a particular category of entities. Properties characterize the concepts, representing existent relationships among them. The Ontologies can also specify axioms in order to impose restriction (facts that must hold) in the interpretation of elements.

In the Computer Science field, ontologies are commonly employed in AI systems. However, the ontologies cover topics including philosophy, metaphysics, knowledge representation formalisms, development methodologies, knowledge sharing and reuse, knowledge management, business process modeling, commonsense knowledge, systematization of domain knowledge, information retrieval from Internet, evaluation and standardization (DEVEDZIC, 2002).

2.2.1 Ontology Languages

2.2.1.1 Frames

The frame theory was proposed by Minsky (MINSKY, 1974) aiming to semantically direct the reasoning. Minsky uses a scene-analysis system to illustrate his frame theory. However, several posterior works employ the frames just to represent a structure to store information, developing the reasoning separately. According to Minsky, the theory essence is on those circumstances where we select from our memory a structure called frame, which is adapted complying the reality.

A frame consists of a data structure to represent stereotyped situations. It contains several types of information, part of them being about how to manipulate that frame. It also contains information about the expectations related to the future frames, or even about how to react if the expectations do not become true. Frames usually represent typical situations, based on the idea that the conceptual codifications done in the brain is related with the most evident properties of objects. There is not a concern in defining exhaustively the all properties of the objects.

Frames are identified by a name and they contain a set of slots. A slot represents a property and it is composed by attributes called facets. Each facet imposes semantic constraints in the values that the slot can assume. Facets can specify the slot datatype, domain of possible values, default values, frames to describe the slot or procedures to determine the slot value. The Figure 2.4 illustrates an abstract representation of a frame and the example representing a Car.

The abstract frame representation is composed by a frame name and two slots. The Car frame is composed by two slots: number of wheels and category. The former slot is numeric and its default value is 4. The last slot is a string and it can assume the values sedan, pickup or hatchback.

The inheritance of frames is obtained through a specific inheritance slot. Specialized frames inherit properties from their parents, including slots, default values and methods.

2.2.1.2 DAML+OIL

DAML+OIL (Darpa Agent Markup Language + Ontology Interchange Language) is a semantic markup language for Web resources. It was developed based on the RDF and RDF Schema, and it extends those languages with primitives for a richer modeling (CONNOLLY et al., 2001). DAML was developed by a DARPA (Defense Advanced Research Projects Agency) project called DAML-ONT, which aimed to improve the accessibility and the interpretation of data on the Web context. The DAML-ONT was developed simultaneously with the OIL (Ontology Inference Layer) (FENSEL et al., 2000), which had similar objectives. The DAML-OIL resulted from the integration of those efforts.

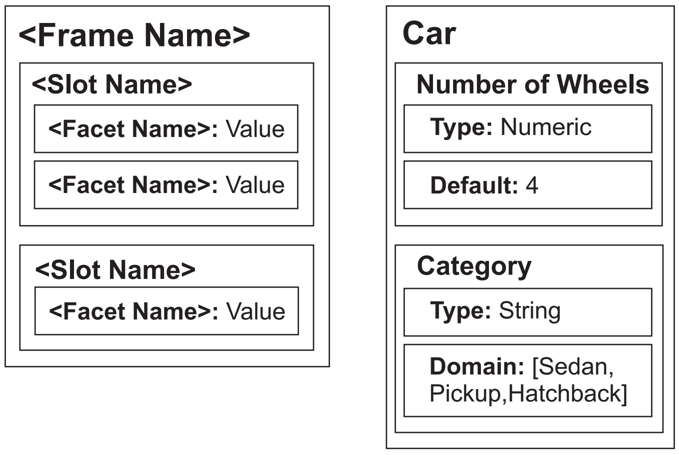


Figure 2.4: Frame abstract representation (left) and Car frame example (right).

This language was built as an additional layer over the RDF, using the same XML syntax, however, providing extensions such as cardinality constraints, transitivity, inverse and uniqueness of property values, and the capability of expressing disjoint classes. Basically, the DAML+OIL was developed to create ontologies in such way that the information can be logically computed. The DAML+OIL is not a W3C (World Wide Web Consortium) recommendation, but it has contributed significantly to the OWL development.

2.2.1.3 RDF and RDF Schema

Resource Description Framework (MANOLA; MILLER, 2004) is a language to represent information about resources on the World Wide Web. More accurately, it was built to express metadata about Web resources, such as title, author, copyright, licensing of documents, or availability of a particular shared resource. Generalizing the concepts of Web resource, the applicability of the RDF increases significantly since that language can be used to express information about anything that can be identified in the Web.

According to (CARROLL; KLYNE, 2004), the RDF development was motivated by the following usages, among others:

- Web metadata: providing information about Web resources and the systems that use them.
- Applications that require open information models.
- To allow data to be processed outside the particular environment in which it was created, in a fashion that can work at Internet scale.
- Automated processing of Web information by software agents: the Web is moving from having just human-readable information to being a world-wide network of cooperating processes.

RDF intends to represent information in a flexible and minimally constraining way. Following the Semantic Web goals, the RDF is directed towards circumstances where information has to be used by software applications instead of simply displayed to humans. The RDF language provides a common framework to express and share such information among heterogeneous applications without loss of semantics.

The RDF syntax is a set of triples, called RDF graph. The triples are composed by:

- The subject, which is a RDF URI (Uniform Resource Identifier) reference or a blank node.
- The predicate (property), which is a RDF URI reference.
- The object, which is a RDF URI reference, a literal or a blank node.

To store the graphs it is used a syntax based on XML, called RDF/XML. This syntax, like HTML (Hypertext Markup Language), is machine processable, and using URIs it can connect pieces of information across the Web. However, unlike the HTML, the RDF URIs can refer to any identifiable thing, even those that may not be directly retrievable on the Web. The RDF can be viewed as a XML application to represent metadata, in other words, it provides a standard way to represent metadata using XML. This language has not primitives to declare classes, properties and relationships. Those primitives are specified by the RDF Schema.

The RDFS (RDF Schema) does not provide a vocabulary of application-specific classes. Instead, it provides the facilities need to describe such classes and properties. In other words, it provides a type system for RDF. The schemas (vocabulary descriptions) written in RDFS are valid RDF graphs. By this it means that an application that was not developed to process the RDFS additional vocabulary still interpret those schemas as valid ones. However, that software application is incapable of interpreting the meaning of the pre-defined RDFS terms (BRICKLEY; GUHA, 2004).

The RDFS can be considered an ontological language since it defines the semantics for classes and properties, constraints, and inheritance. However, this language is very limited in the sense that it does not provide an inference mechanism, and it needs to be improved to describe information in a more detailed way. The RDFS was taken as base for the OIL, DAML+OIL and OWL.

2.2.1.4 OWL

Web Ontology Language (OWL) (SMITH; WELTY; MCGUINNESS, 2004) is intended to provide a language to specify the classes and relations between them in Web documents and applications. This language is part of an effort of W3C to make the Semantic Web possible.

The OWL classes provide a way to group resources with similar characteristics. The classes are associated with a set of individuals, called class extension. The individuals are called instances of the class. Class descriptions correspond to basic building blocks of class axioms. It describes an OWL class either by a class name or by specifying the class extension of an unnamed anonymous class. The OWL language differentiates the following types of class descriptions (SMITH; WELTY; MCGUINNESS, 2004):

- A class identifier: it describes a class through a class name, syntactically represented as a URI reference. It is a subclass of *rdfs:Class*.
- An exhaustive enumeration of individuals that together form the instances of a class. It is defined with the *owl:oneOf* property.
- A property constraint is a special type of class description. It describes an anonymous class, namely a class of all individuals that satisfy the restriction. There are two kinds of property constraints: value constraints and cardinality constraints.

- Intersection, union and complement of two or more class descriptions. They represent the more advanced class constructors that are used in Description Logic. They can be viewed as representing the *and*, *or* and *not* operators on classes.

OWL properties can be grouped in two main categories: object properties, which connect individuals to individuals, and datatype properties, which link individuals to values. Property axioms define the characteristics of a property.

Individuals are described using individual axioms, also known as facts. The OWL considers two kinds of facts: those about class membership and property values of individuals, and those facts about individual identity. On the Web, the unique names (different names refer to different things) assumptions is not possible. Therefore, the OWL provides three constructs for stating facts about the identity of individuals: *owl:sameAs*, *owl:differentFrom* and *owl:allDifferent*.

This Section does not intend to be an exhaustive description of the OWL features. Further information about the OWL syntax, semantics and other related issue, can be found in (SMITH; WELTY; MCGUINNESS, 2004).

According Horrocks (HORROCKS; PATEL-SCHNEIDER; HARMELEN, 2003), the OWL uses the fact-stating ability of the RDF and the structure of classes and properties capabilities from RDFS, and it extends them in important ways. Like RDFS, the OWL also can declare classes, and organize these classes in a subsumption hierarchy. The classes can be specified as logical combinations (intersections, unions, or complements) of other classes, or as enumerations of specified objects, going beyond the capabilities of RDFS. The OWL can also declare properties and subproperties (a hierarchical organization like the classes and subclasses). The domains of OWL properties are OWL classes, and ranges can be either OWL classes or externally-defined datatypes such as string or integer. OWL can state that a property is transitive, symmetric, functional, or is the inverse of another property, here again extending RDFS.

There are three OWL sublanguages. They present different levels of expressiveness, and they are intended to be used by different community of users and developers. They are:

- OWL Lite: designed to those users needing a classification hierarchy and simple constraint features. Through this improvement on tractability, the OWL Lite provides a simpler and quick migration for thesauri and other taxonomies. However, that improvement comes with relatively loss in the expressive power. In example, the OWL Lite only allows cardinality values of 0 or 1. The OWL Lite is a subset of the OWL DL.

- OWL DL: designed to those users who want the maximum expressiveness, without losing computational completeness and decidability of reasoning systems. According to (HORROCKS; PATEL-SCHNEIDER; HARMELEN, 2003), the OWL DL is an extension of a restricted use of RDF and RDFS, because, unlike RDF and RDFS, they do not allow classes to be used as individuals, and the language constructors cannot be applied to the language itself. OWL DL was designed to support the existing Description Logic segment and it has desirable computational properties for reasoning systems.

- OWL Full: it provides the full expressiveness and the syntactic freedom of RDF, however, with no completeness and decidability guarantees. The OWL Full contains the OWL DL. Horrocks (HORROCKS; PATEL-SCHNEIDER; HARMELEN,

2003) claims that the penalty for that is two-fold. First, reasoning in OWL Full is not decidable, since the constraints required to maintain the decidability of OWL DL do not apply to OWL Full. Second, the abstract syntax for OWL DL is inadequate for OWL Full, and the official OWL exchange syntax, RDF/XML, must be used. The OWL Full allows an ontology to improve the meaning of the pre-defined RDF and OWL vocabulary. However, reasoning systems developed to those languages will not be able to perform inferences considering the expanded vocabulary.

Despite the fact that OWL is a recent language, its development is the result of OIL, DAML-ONT and DAML+OIL efforts. The OWL is maintained by the W3C, which specified the language to be used in the Semantic Web (BERNERS-LEE; HENDLER; LASSILA, 2001).

2.2.2 Building Ontologies

Uschold and Gruninger (USCHOLD; GRUNINGER, 1996) claim that there is no standard methodology for building ontologies. In an attempt to fill this gap, the authors propose a skeletal methodology for building ontologies. It is composed by the following stages:

- Identify Purpose and Scope: it is fundamental to be clear about why the ontology is being developed and what its intended uses are.

- Building the Ontology: it is composed by three stages. The first begins by capturing the ontology, what means: identification of the key concepts and relationships in the domain of interest, production of precise unambiguous text definitions for such concepts and relationships, selection of terms to refer to such concepts and relationships, and finally, agreeing on all of the above. The second stage is the ontology coding, which involve committing to the basic terms that will be used to construct the ontology, selecting a language to codify the ontology, and writing the code. The last stage consists in integrating the ontology with existing ones.

- Evaluation: to check if the ontology addresses its requirements (competency questions, requirements specifications) it is necessary to make a technical judgment.

- Documentation: an adequate documentation allows the reuse and integration of existing ontologies. Concepts, relationships and other entities important to provide the ontology understanding should be documented according to the type and purpose of the ontology.

Ontological engineering encompasses the tasks conducted during conceptualization, design, implementation and deployment of ontologies. A discussion about ontological engineering is presented in (DEVEDZIC, 2002). This work also intends to cover the relations among the ontological engineering and other disciplines.

2.3 BDI Agent Architecture

Franklin and Graesser (FRANKLIN; GRAESSER, 1996) review and discuss agent definitions proposed by several researchers. It becomes clear that there is no general agreement about what constitutes an agent or what differs agents from programs. In order to capture the essence of being an agent and clearly distinguish an agent from an ordinary program, the authors propose a formal definition of an autonomous agent: an autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

According to Wooldridge and Jennings (WOOLDRIDGE; JENNINGS, 1995), it is possible to differentiate two general usages for the term agent. The first is weak, and relatively uncontentious. The second is stronger, and potentially more contentious. In the weak notion of agency, the term agent denotes a hardware or software system that exhibits:

- Autonomy: agents operate without intervention of humans and control their behavior and internal state.
- Social Ability: agents interact with other agents and their environment in order to achieve their goals.
- Reactivity: it is a property of agents that perceive their environment and respond in a fashion time when necessary.
- Pro-activeness: agents do not simply respond to environment changes, they take the initiative to achieve their goals.

For some researchers, the agency has a stronger and more specific meaning. In a strong notion, an agent is described as a computing system that exhibits the above mentioned characteristics and also additional properties common to human beings, such as:

- Mobility: mobile agents have the ability to migrate from a platform to another using a computer network infrastructure.
- Benevolence: it is the agent's capability in cooperating with other agents.
- Rationality: rational agents always choose the best alternative available in order to achieve its goals.
- Veracity: it means that agents will not intentionally communicate false information.
- Adaptability: adaptable agents learn through experiences and consequently change their behavior to adapt to dynamic environments.

The agent-oriented paradigm can be applied to several contexts, including those that involve distributed control, distributed data management, complex problems that have to be decomposed, heterogeneity of knowledge representations, multiple functionalities to be carried through by the system, and some degree of system autonomy.

The taxonomy presented by (BRENNER; ZARNEKOW; WITTIG, 1998) classifies agents taking into account their functionalities: information agents, cooperation agents, transaction agents. Information agents search for particular information to their owners in a distributed system or computer network. They extract information from sources, filter results and present the result in an adequate way. The cooperation agents focus on the resolution of complex problems through communication and coordination. They are employed in circumstances that demand synergy. Finally, the transaction agents monitor and process transactions. The usage of this kind of agent guarantees a higher security, robustness and trustiness to the transaction.

However, that perspective does not fit to most cases, being necessary to classify the agents by their processing strategy (internal architecture), independently of the role played by them. The agent architecture specifies how this agent can be decomposed in modules, and how these modules interact. The modules and their interactions describe how the perceptions and the agent's internal state determine the actions (MAES, 1995). Usually, agents are classified in reactive or cognitive, but some authors also employ the notion of hybrid architecture to classify those that mix characteristics of both.

Reactive agents have competency modules, which allow the response to particular events (BROOKS, 1986). Agents of this class tend to be structurally simple since they do not have an explicit representation of its environment, and they are not capable to perform sophisticated logic reasoning. They made their decisions based on the current situation since no history is stored. Their intelligent behavior is obtained through the interaction with the environment and other agents. The reactive agents are closely related to the Swarm Intelligence area, which designs algorithms and techniques for distributed problem resolution based on the collective behavior exhibited by the social insects and other animal societies.

Cognitive agents, also known as deliberative agents, have an explicit and symbolic representation of their environment. They keep the history of their internal states and actions, what enables to consider past decisions in the future deliberations. Generally, they take their decisions through logic reasoning on the knowledge base. Most cognitive architectures are founded on the human practical reasoning, which is explained through the ascription of mental states to the agents. The human practical reasoning consists in choosing between competing alternatives, where relevant considerations are determined by what the agent believes, values, and cares about (WOOLDRIDGE, 2000). It is important to distinguish practical reasoning from theoretical reasoning. The former is directed towards action, and the last is directed towards beliefs. The human practical reasoning is composed by at least two distinct processes: deliberation, which involves deciding what states of affairs the agent will pursue, and means-end reasoning, which defines how the agent will achieve the states selected in the deliberation process.

Architectures based on mental states adopt a psychological perspective to define the structures of the agents: such as beliefs, desires, intentions, expectations, capabilities, commitments and choices. The BDI model (BRATMAN; ISRAEL; POLLACK, 1988) is probably the most known cognitive agent architecture. The name received by this model is justified by its three mental states: belief, desire and intention. It recognizes the primacy of those three mental states in rational action.

2.3.1 Mental States

According to the folk psychology, the human behavior can be predicted and explained through the ascription of mental states, such as beliefs, desires and intentions. The main idea behind of this approach is that cognitive agents have internal states that relate to the state of the agents' environment. These states have a correspondence with the human mental states in terms of significance and existence (CORRêA, 1994).

McCarthy (MCCARTHY, 1979) claims that to ascribe beliefs, abilities, intentions, free will, or wants to a machine is legitimate when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps to understand the machine structure, its past and future behavior, or how repair or improve it.

There are other reasons, beyond the high abstraction level, to believe that the agent modeling as intentional system is useful to understand computer programs. First, maybe the most important, is the communication ability among heterogeneous agents, which results in the ability of communicating mental states. Second, the mentalist models can be employed to represent information about final users.

Mental states can be categorized in informational or proactive. Informational mental states are related to the information that the agent has about its world (i.e. beliefs). The proactive mental states tend to lead agents to action (i.e. desires and intentions).

Beliefs represent the agents' knowledge about their world. From a computational viewpoint, beliefs are a way to represent the state of the word, through variables, a relational database or symbolic expressions. Beliefs are fundamental because the world is dynamic (past events need to be remembered), and the systems have just a local view of that world (GEORGEFF et al., 1999). Wooldridge (WOOLDRIDGE, 2000) claims that beliefs correspond to information that agents have about their world, and these information may be incomplete or incorrect.

Desires correspond to states of the world that agents want to bring about. They do not necessarily cause actions. By this it means that before performing any action, the agent will deliberate about which desires it will commit to achieve. Desires have the following characteristics:

- Represent states of affairs that the agent would like to achieve.
- Desires can be inconsistent (mutually exclusive) with other desires. If an agent desires competing desires, it does not mean that will act to reach them.
- Desires do not lead directly to actions, but they can generate their occurrence when they become intentions.

Intentions correspond to states of the world that agents have committed to achieve. They can be considered a subset of desires, since agents will intend only states of affairs they desire. However, differently from desires, intentions have to be consistent with other intentions. They are formed through a deliberative process and other intentions, but agents may have initial intentions defined by the application developer. Generally, the intention notion is employed to characterize both states of mind and intentional action. The intention mental state is directed to actions in the future and it will not necessarily lead to action. The intentional actions are directed to the present and they represent the immediate actions.

Wooldridge (WOOLDRIDGE, 2000) asserts that intentions play the following roles in the practical reasoning:

- Intentions drive the means-end reasoning: once formed an intention, the agent will have to deciding how to achieve the intention; moreover, if a course of action fails, the agent has to attempt others if available.
- Intentions persist: they will persist until the agent believes it has successfully achieved them, it believes it is impossible to achieve them, or it believes the reason for the intention is no longer present.
- Intentions constrain future deliberations: agents will not entertain options that are inconsistent with their current intentions.

Intentions influence beliefs upon which future practical reasoning is based: agents assemble future plans assuming that the current intentions will be successful. Having an intention, while believing that it will not bring about is named intention-belief inconsistency, and it is not rational. Having an intention without believing that it will be bring about is named belief-intention incompleteness, and it is an acceptable property of agents. The distinction between those two cases is known as asymmetry thesis (BRATMAN, 1987 apud WOOLDRIDGE, 2000).

2.3.2 Reasoning

In the deliberation process, BDI agents examine which desires are possible, choose between competing desires, and commit to achieving them (WOOLDRIDGE, 2000). The deliberation process begins by verifying which desires are possible to be achieved (option generation). Following, it checks the consistency among possible desires and intentions (filtering).

Once formed the intentions, the agent has to assemble action plans to achieve them. This process is known as means-end reasoning. Research on AI has proposed several solutions concerning the planning problem, such as methods for searching the space of possible actions. However, in the cognitive multiagent domain, most applications use a plan library to address the means-end reasoning issue. These libraries are commonly composed by previous assembled plans and the circumstances they can be applied.

When an intention is formed by the deliberation process, the agent has committed to achieve that intention. The commitment strategy implies temporal persistence and it specifies how committed an agent should be to its intentions. The three following commitment strategies are commonly discussed in the BDI literature:

- Blind: intentions will be kept until the agent achieves them.
- Single-Minded: intentions will be kept until the agent achieves them or the agent believes that it is not possible to achieve them.
- Open-Minded: intentions will be kept while the agent believes it is possible to achieve them.

Usually, a cautious agent reconsiders its intentions after the following conditions: achievement of an intention (or impossibility of achieve an intention) and conclusion of a plan execution. Supposing that an agent reconsiders its intentions during the execution of a plan, possibly after each action, it is called audacious (WOOLDRIDGE, 2000).

The intention reconsideration is related to changes on the agent's environment. In an environment that rarely changes, a cautious agent will spend a lot time reconsidering its commitments, while an audacious agent will be busy pursuing its intentions. In dynamic environment where changes happen frequently, cautious agents have better chances than audacious ones to detect opportunities in the environment.

2.3.3 Examples of BDI Architectures

The IRMA (Intelligent Resource-bounded Machine Architecture) (BRATMAN; ISRAEL; POLLACK, 1988) abstract architecture aims the description of the processes involved in the practical reasoning in resource-bounded agents. It can be classified as a BDI model since it includes the representation of beliefs, desires and intentions. The Figure 2.5 illustrates the IRMA architecture.

Intentions are structured inside plans, which can be grouped in two categories:

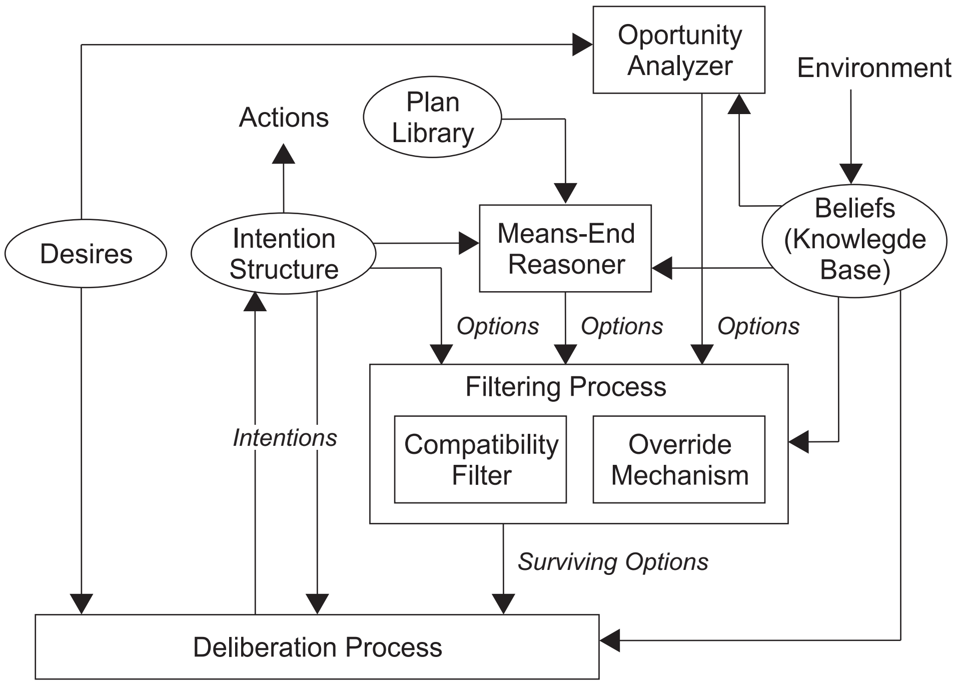


Figure 2.5: IRMA Abstract Architecture.

- Plan Library: corresponds to a subset of agent's beliefs about actions and their effects under specific conditions.

- Intention Structure: corresponds to the agent's current plans.

The IRMA plans can be partial, in other words, it is possible to decide about the state to be pursued, but postpone the decision about how to achieve that state. The means-end reasoner is responsible for the assembling of plans. If the system has already built partial plans, the means-end reasoner tries to complete them. The opportunity analyzer component aims the option proposal in response to environment changes perceived by the agent. The compatibility filter checks if the options are consistent with the current intentions. Options considered inconsistent are forwarded to the override mechanism, which is sensitive to problems and opportunities perceived by the agent. Those perceptions indicate conditions under which plans have to be dropped or checked against other options. The surviving options are forwarded to the deliberation process, which forms the intentions to be introduced inside the plans.

The PRS (Procedural Reasoning System) (GEORGEFF; INGRAND, 1989) is a hybrid architecture that considers the BDI mental states. The PRS is probably the most known BDI system. It was implemented by several researchers (RAO, 1996; D'INVERNO et al'., 1998; HUBER, 1999). Its internal structure is composed by the following components: database (beliefs), goals, intention structure, KA (knowledge area) library (plans) and interpreter (reasoner). The Figure 2.6 depicts the PRS architecture.

The database stores the beliefs which are specified using first-order logic. The goals correspond to the current desires. Agent can have goals related to the current intentions, named sub-goals, and goals that have not relation with current intentions. The KA library contains the plans, which are composed by a body (sequence of actions) and the conditions under which the plan fits. It is possible to specify partial plans. The intention structure contains the tasks (plans) to be executed by the agent. The reasoner manages the other components, selecting the plans taking into account the beliefs and goals, and putting the selected plans inside the intention structure to be executed.

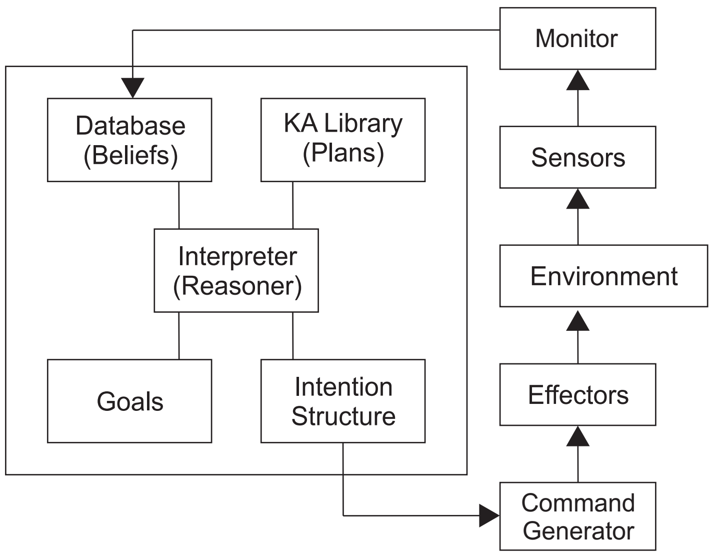


Figure 2.6: PRS architecture.

The InteRRaP (MÜLLER, 1996), illustrated in the Figure 2.7, is a layered BDI architecture, whose goals consist in treat unexpected events, react in a fashion time, and exhibit goal-oriented behavior and multiagent coordination abilities. These requirements are fulfilled by integrating reactivity, deliberations, interaction and cooperation is the same architecture. According to the author, previous approaches do not address these requirements, focusing just in a subset of them.

The InteRRaP architecture is divided in three layers:

- Behavior: reactivity and procedural knowledge.
- Local Planning: goal-oriented behavior and means-end reasoning.
- Cooperative Planning: reasoning about other agents, and coordinated actions.

To guide the InteRRaP development, it was taken into account design decisions such as layered control and layered knowledge base. It enabled the different abstractions levels and the storage of beliefs in a hierarchic way. The layer activation is bottom-up. If a layer is not capable of dealing with the situation, it forwards the control to the next layer. The action execution is top-down. The higher layers use primitives defined by the lower ones.

2.3.4 Interoperability for Agents

Efforts concerning interoperability for cognitive agents have been done mostly at the syntactic level. However, it is desirable to achieve also the semantic interoperability. In the multiagent context, it consists in exchanging information between agents, ensuring that the receiving agent will interpret accurately the meaning of that information.

The KQML (Knowledge Query Manipulation Language) (FININ et al., 1994) supports the social interaction between agents by providing an ACL (Agent Communication Language) based on the speech acts theory. However, it is not considered a standard since there is not an agreement among the KQML community about the specification of

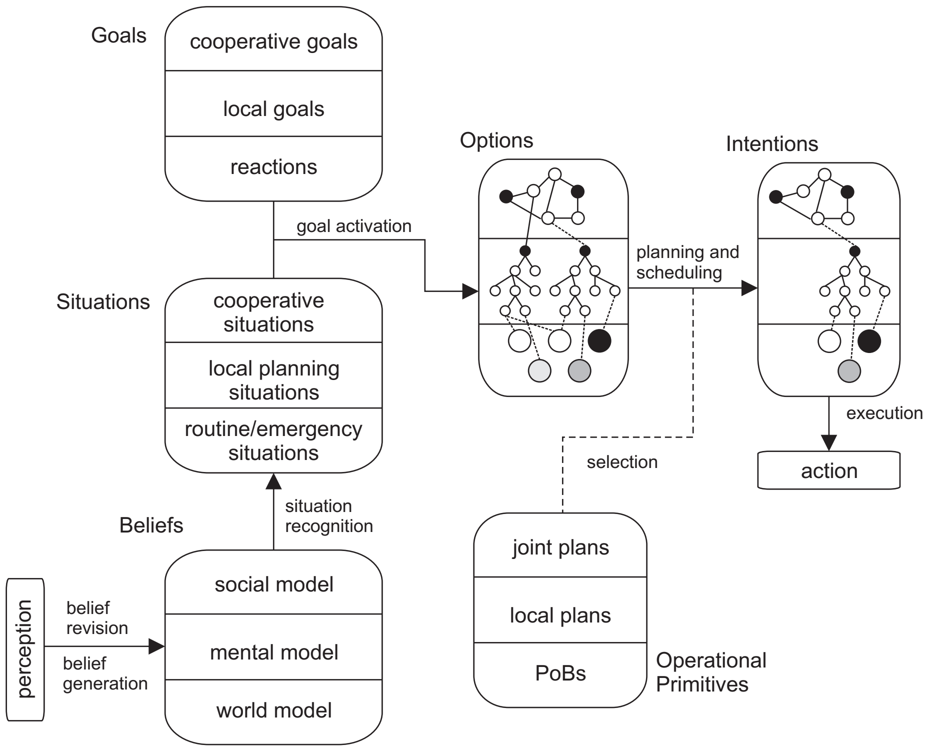


Figure 2.7: InteRRaP architecture.

the language. Currently, there are several KQML dialects, what goes against the desired interoperability.

The FIPA (Foundation for Intelligent Physical Agents) (FIPA: Foundation For Intelligent Physical Agents, 2006) is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. To promote the communication among agents that organization proposed the FIPA-ACL (FIPA: Foundation For Intelligent Physical Agents, 2002). The structure of this language has resemblances with the KQML since it is also founded on the speech acts theory. A FIPA-ACL message is composed by parameters, including performative (speech act), sender, receiver, content, language and ontology.

Differently from KQML and FIPA, the OMG (Object Management Group) MASIF (Mobile Agent System Interoperability Facility) effort focuses on the agents' mobility. The KQML and FIPA point out that the social interaction is the most fundamental feature of a multiagent system. The MASIF does not approach interoperability among agents of different platforms. The interoperability proposed by that initiative is limited to the CORBA (Common Object Request Broker Architecture) architecture, and it also does not treat semantic interoperability.

Both KQML and FIPA standards do not treat the semantics of the message content. However, they contain a parameter to indicate the ontology used to specify the concepts present in the message content. Thus, the agent has to adopt an language, such as OWL, to specify the ontology. Through the ontology, the semantic interoperability is obtained.

3 AN ONTOLOGY-BASED APPROACH TO REPRESENT BAYESIAN NETWORKS

One of the contributions of this research is the specification of an ontology to formalize the Bayesian network knowledge representation. This ontology remodels and extends the concepts defined in (SANTOS; BOFF; VICARI, 2006), allowing a broader utilization of it. In the next Chapter is presented one of these utilizations, an ontology-based approach to interoperability for cognitive Bayesian agents.

To guide the ontology development, it was adopted the methodology proposed by Uschold and Gruninger (USCHOLD; GRUNINGER, 1996), detailed in the Section 2.2.3. The organization of this Chapter has a correspondence to the methodology stages. The Section 3.1 aims the description of the purpose and the scope of the ontology. The conceptualization capture is detailed in the Section 3.2. The explicit representation of the captured concepts in OWL are presented in the Section 3.3. The Chapter is closed by the final considerations in the Section 3.4.

3.1 Ontology Purpose and Scope

The ontology purpose is to enable the interoperability and the reusability of the Bayesian network knowledge representation. In this work, the application scope is restricted to a cognitive agent architecture, where the ontology addresses interoperability issues and representation of agent's knowledge. Intended users are developers and researchers that need to share common concepts about Bayesian networks. Users that desire to build models of other probabilistic networks are also expected, since they can do it by extending the proposed conceptualization.

3.2 Ontology Capture

This Section aims the identification of concepts and relationships in the domain of discrete Bayesian networks, and their representation independently of implementation language. The capture process is supported by the Bayesian networks theoretical foundations presented in the Section 2.1.

The conceptual maps were adopted to graphically represent the concepts, since they provide a straightforward way to visualize classes and their relationships. The classes are represented through rectangles and the relationships are represented through labeled arrows. A special kind of relationship, the inheritance, is represented through the *isA* arrow. In an *isA* relationship, the child class inherits all father's relationships. The following conceptual maps hide relationships already depicted in a previous map. The semantic restrictions of properties are specified in natural language.

In the Section 3.2.1 it is presented the graph representation. It defines the common concepts among different probabilistic networks. The Section 3.2.2 presents the discrete Bayesian network ontology. The concepts related to evidences and their relation with the evolution of Bayesian network individuals are presented in the Section 3.2.3. Finally, the Section 3.2.4 is concerned with the ontology extensions.

3.2.1 Graph Representation

Probabilistic networks are graphical models of causal interactions among a set of variables, where the variables are represented as nodes of a graph and the interactions as directed arcs between nodes (COWELL et al., 1999). A graph is the basic structure shared between probabilistic network models. It is formalized in the ontology by the Graph class (Figure 3.1). It has two properties named *hasNode* and *hasArc*, which respectively are links to multiple individuals of classes *Node* and *Arc*. These two properties represent elementary components of a graph. It is considered that a graph has at least one node. Such cardinality constraint is imposed in the *Graph* property *hasNode*.

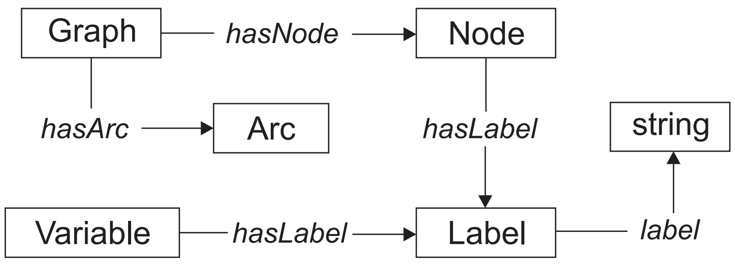


Figure 3.1: Graph representation.

The *DirectedGraph* class (Figure 3.2) is a subclass of *Graph* that represents a directed graph. This graphical model is common to variations of probabilistic network knowledge representation (i.e. Bayesian networks, influence diagrams and object-oriented probabilistic networks). The common elements are the directed arcs and the nodes, respectively indicated by the inherited properties *hasArc* and *hasNode*. The *hasArc* property is semantically restricted to link *DirectedArc* class individuals.

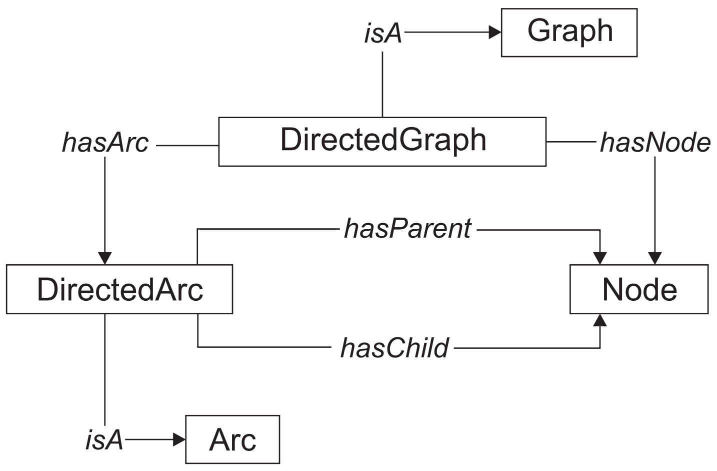


Figure 3.2: Directed Graph representation.

To represent a directed edge between a parent and a child node it is defined the *DirectedArc* class, a specialization of the *Arc* class. The *DirectedArc* has two properties: *hasChild* and *hasParent*. The value of these properties is a single individual of the *Node* class. This class is defined by a unique label, denoted by the *hasLabel* property. The label, a common attribute among chance, decision and utility nodes, is a string datatype. Since this work deals with Bayesian networks it was specified only chance nodes.

Another general concept concerning probabilistic networks is defined by the *Variable* class. It represents a set of mutually exclusive states. The states, also called events or choices, correspond to the domain of the variable, which can be discrete or continuous. In this work it is considered only discrete variables (finite sets). A probabilistic network has two categories of variable: chance variables, representing random states, and decision variables, representing choices controlled by some agent. As the *Node* class, the *Variable* has only the *hasLabel* property. The purposes of the *Variable* class are the same of the *Node* class, which are to abstract the complexity and model common aspects between its subclasses.

3.2.2 Discrete Bayesian Network Representation

A discrete Bayesian network consists of a DAG (Directed Acyclic Graph) and a set of conditional probability distributions (JENSEN, 2001). Each node in the network, called chance node, corresponds to exactly one discrete random variable which has a finite set of mutually exclusive states. The directed arcs specify the causal relation between the random variables. Each random variable associated with a chance node has a conditional probability distribution.

The *BayesianNetwork* class (Figure 3.3) is the core of the Bayesian network definition. It is a subclass of *DirectedGraph*. The differences among these classes are the semantic constraints imposed to the properties to specify the correct type of nodes and arcs allowed in a Bayesian network. Such kinds of nodes and arcs are represented by the *ChanceNode* and *BayesianArc* classes respectively.

A *BayesianArc* individual defines an edge between two chance nodes. It inherits the *hasParent* and *hasChild* properties from the *DirectedArc* class, and imposes additional constraints to formalize that only individuals of the *ChanceNode* class can be assigned to these properties. A *ChanceNode* individual has a chance variable associated to its definition. Such variable is indicated by the *hasChanceVariable* property. This property allows only individuals of *PriorChanceVariable* and *ConditionalChanceVariable* classes. Such constraint is necessary to differentiate variables of prior nodes from variables of non-prior nodes. The Figure 3.4 represents the Prior Chance Variable and the Figure 3.5 represents the Conditional Chance Variable.

Before defining a chance variable it is necessary to define a state and its related concepts. A state is represented by the *State* class, which has only the *hasLabel* property responsible for the node identification. The *State* class has two direct subclasses. The first, called *StateProbability*, denotes a chance associated with a state. It has two properties: the inherited *hasLabel* and the probability (float data type). The second specialization of *State* is named *ConditionalState* and it specifies the multiple conditional chances associated with a state. A set of *ConditionalState* individuals, mutually exclusives, constitutes a Conditional Probability Table (CPT). The *ConditionalState* class has two properties: the inherited *hasLabel*, which represents the label of the state and the *hasConditionalProbability* property, which indicates multiple individuals of the *ConditionalProbability* class.

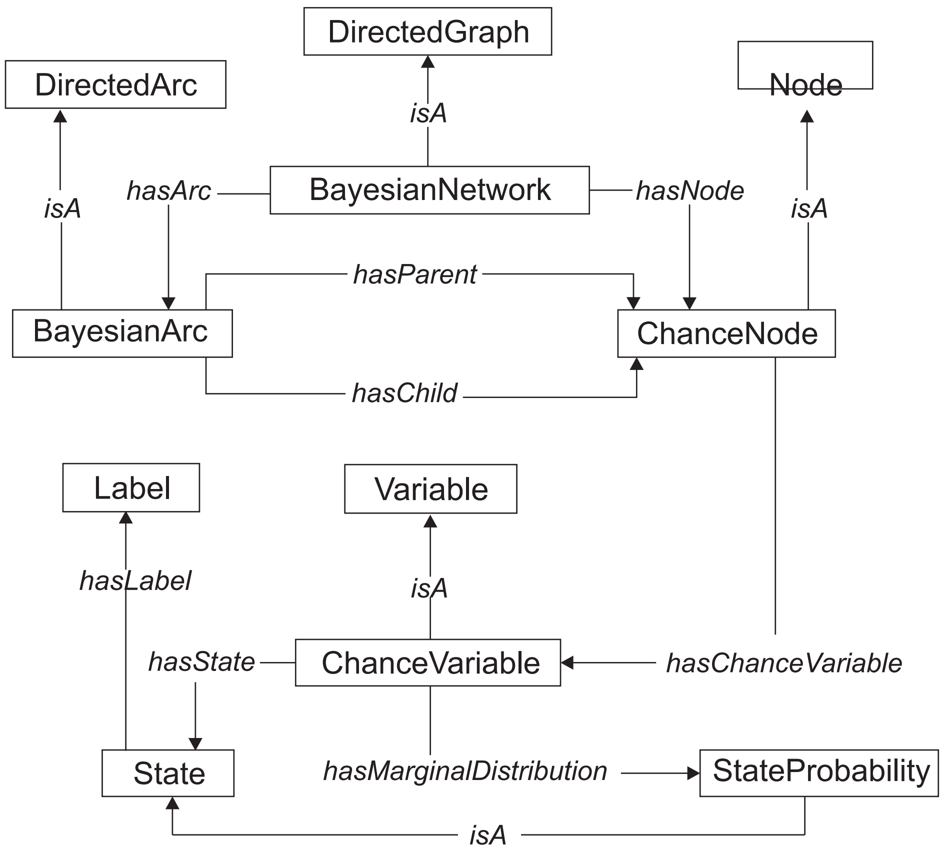


Figure 3.3: Bayesian network representation.

The *ConditionalProbability* class represents the conditional chances associated with a state. This class has two properties: *probability* and *hasCondition*. The former is a float data type property that represents the numerical probability of a variable's state under the conditions specified in the *hasCondition* property. This property indicates multiple individuals of the *Condition* class, and it denotes the conditions imposed in the probability of a state. The *Condition* class is constituted by a conditioning node and a state of this node, respectively indicated by the properties *hasNode* and *hasState*. The individual indicated by the *hasNode* property must be a *ChanceNode* since only chance nodes have random variables. The *hasState* property indicates an individual of the *State* class that represents the specific state of the conditioning chance variable.

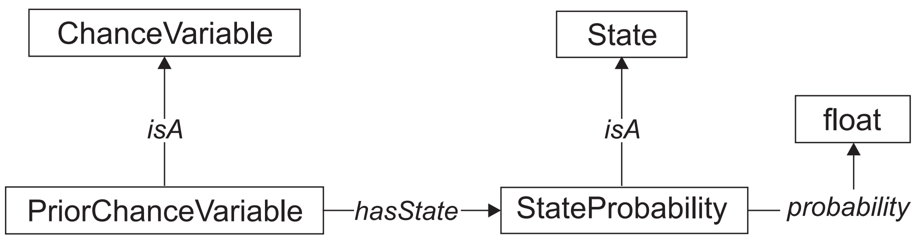


Figure 3.4: Prior Chance Variable representation.

The *ChanceVariable* class is a specialization of *Variable* and it represents a chance variable. Additionally to the inherited properties from the *Variable* class, it specifies the *hasState* and *hasMarginalDistribution* properties. The first property indicates the necessity of at least one state (*State* individuals) associated with a variable (i.e. true or false in the context of a Boolean variable). The second property indicates the computed marginal distribution for the chance variable, and it is restricted to multiple *StateProbability* individuals. Each individual represents a state and its computed chance of occurrence. The marginal distribution is optional in the model since it can be computed using the CPT.

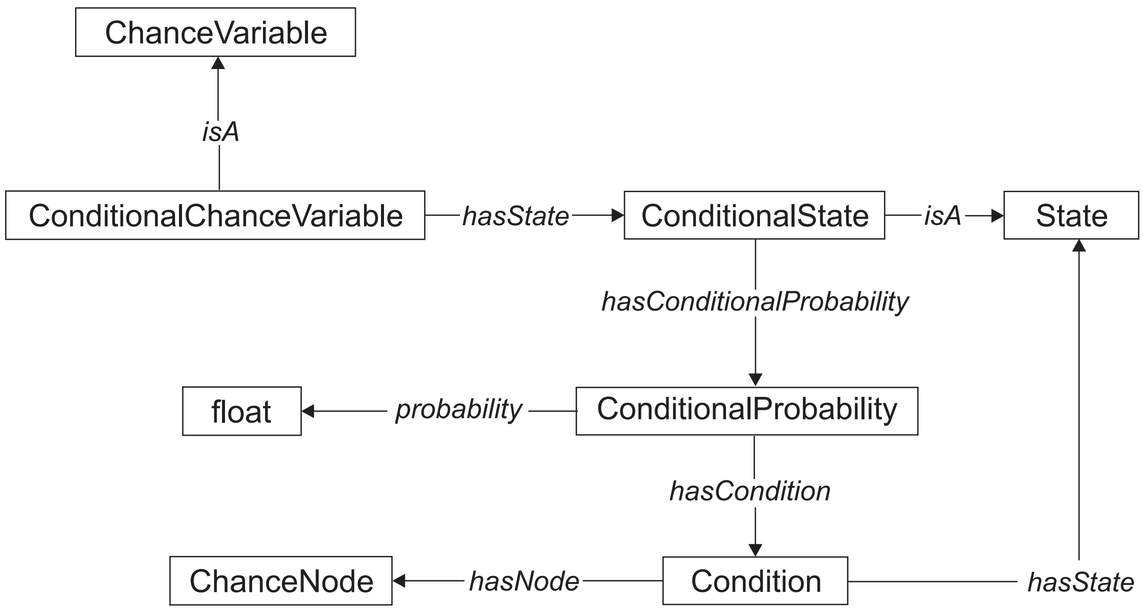


Figure 3.5: Conditional Chance Variable representation.

It was necessary to differentiate prior node variables from non-prior ones, since a non-prior node has a CPT, and a prior node has only states and probabilities without conditioning variables. Thus, the classes *PriorChanceVariable* and *ConditionalChance-Variable* were created as subclasses of *ChanceVariable*. The difference between these two subclasses lies in the *hasState* property constraint. In the *PriorChanceVariable* class the *hasState* property has been restricted to *StateProbability* individuals. As stated earlier, a state probability represents a state and its chance of occurrence. The *StateProbability* individuals indicated by *hasState* of *PriorChanceVariable* indicate all possible states associated with a prior chance variable. The *hasState* property of *ConditionalChanceVariable* class has also been constrained in way that only *ConditionalState* individuals can be assigned to it. The *ConditionalState* individuals represent a Conditional Probability Table of a variable associated with a non-prior node.

3.2.3 Bayesian Network Situation Representation

In this work, a situation is considered a particular configuration that a probabilistic network assumes given a set (possibly empty) of evidences. These evidences are used in the inference and the result of this process is a new Bayesian network situation. Such situations are useful to keep the history of modifications of a Bayesian network. The Figure 3.6 depicts the situation related concepts.

An evidence, represented by the *Evidence* class, corresponds to any information regarding the state of a variable from a probabilistic network. The *Evidence* class is composed by a node, a label and a chance, represented by the properties *hasNode*, *hasLabel* and *probability*, respectively. The *hasNode* property can indicate only individuals of the *ChanceNode* class, since chance nodes are the only kind of node that represent random events. In order to specify a hard evidence (an observation of an event), it was specialized the *Evidence* class creating the *HardEvidence* class. This class specifies a constraint defining that the probability property must assume the numeric value one.

A situation, represented by the *Situation* class, has two properties. The first property is the *hasGraph*, used to indicate a graph individual which configuration corresponds to the given situation. The second property is the *hasEvidence* that indicates the set of evidences that originates the situation. A particular kind of situation is represented by the *BayesianSituation* class. Its inherited *hasGraph* property can indicate only Bayesian networks.

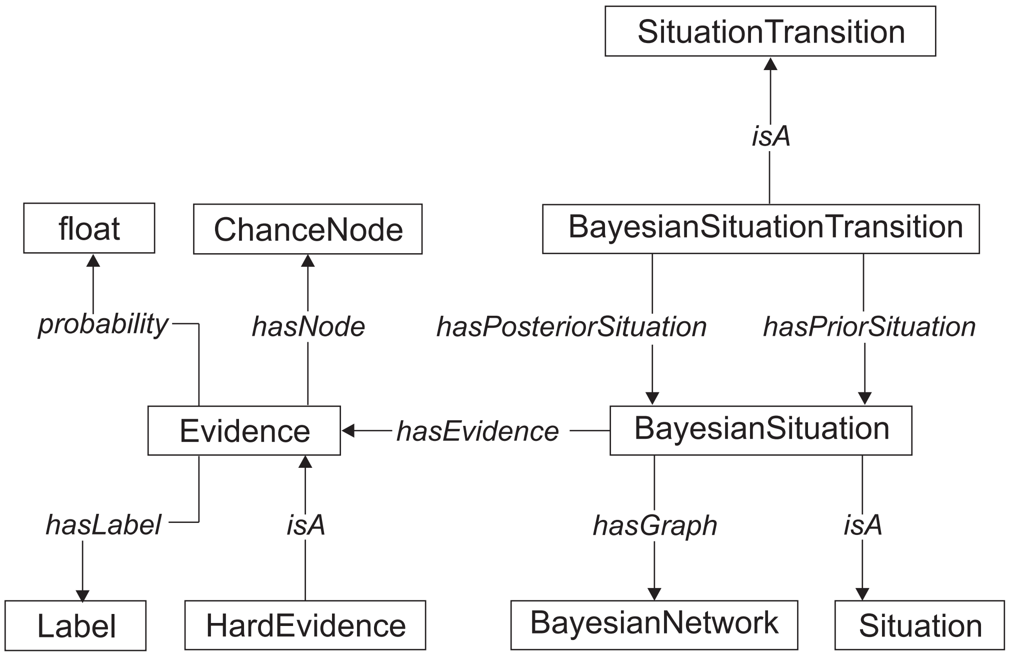


Figure 3.6: Bayesian Network Situation representation.

In order to establish a link between two sequential situations it was created a class named *SituationTransition*. This class has two properties: *hasPriorSituation* and *hasPosteriorSituation*. They respectively represent a prior and a posterior situation. A situation transition between Bayesian networks is represented in the class *BayesianSituationTransition*. This class inherits the properties from *SituationTransition*, constraining them to link only *BayesianSituation* individuals.

3.2.4 Ontology Extensions

The purpose of this Section is to present in a summarized way some possible extensions of the ontology. Graphical representations, such undirected graphs, junction trees, influence diagrams and decision trees, are following discussed. The undirected graphs and the junction trees are better detailed since they are already implemented.

The ontology does not define how the Bayesian network model is executed. A step towards it consists in defining the intermediary structures used in the belief updating (inference). Jensen (JENSEN, 2001) presents an algorithm for belief updating that uses undirected graphs and junction trees (Figure 3.7). To represent the undirected graphs it was created a *Graph* subclass, named *UndirectedGraph*. It has two inherited properties: *hasNode* and *hasArc*. The *hasArc* property is semantically constrained to indicate only *UndirectedArc* class individuals. The *UndirectedArc* class has only the *hasNode* property, which indicates exactly two *Node* individuals.

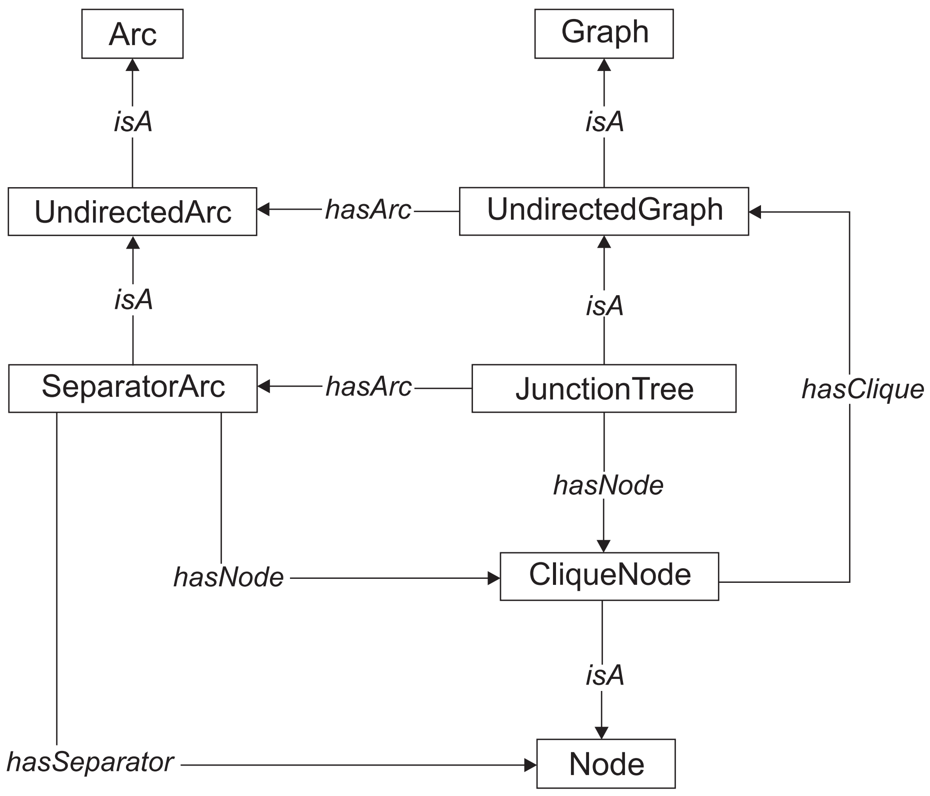


Figure 3.7: Undirected Graph and Junction Tree representations.

Junction trees are undirected graphs composed by a set of cliques and a set of separators. The cliques are also undirected graphs. The separators correspond to the common nodes between two cliques. In order to represent junction trees it was created an *UndirectedGraph* subclass, named *JunctionTree*. It inherits two properties: *hasNode* and *hasArc*. The *hasNode* property is constrained to at least one individual of *CliqueNode* class. The *hasArc* property is restricted to *SeparatorArc* individuals. The *CliqueNode* class is a subclass of *Node* and has the *hasClique* property, which is restricted to one *UndirectedGraph* individual. The *SeparatorArc* class is an *UndirectedArc* subclass that has the *hasSeparator* property, which is constrained to at least one *Node* individual. The inherited *hasNode* property of *SeparatorArc* is restricted to *CliqueNode* individuals.

3.3 Ontology Coding

As stated earlier, the described ontology was implemented using the Web Ontology Language (OWL). The development of the OWL ontology used most of the available functionality of OWL. Our concepts were represented as classes, the *isA* relationship becomes a class specialization, and the remaining relationships were implemented as OWL properties - with the necessary constraints. The utilization of constraints in OWL makes possible the contextualization of concepts and helps to avoid ambiguities.

A snippet of the OWL source code corresponding to the Bayesian network ontology is illustrated in the Figure 3.8. It begins by specifying the *ChanceVariable* class. The next tag contains the comment about the *ChanceVariable* class. Following, it specifies the *hasState* property, with a restriction denoting the minimum cardinality. Finally, it is specified that the *ChanceVariable* class is a subclass of *Variable*. The label of the *ChanceVariable* class is inherited from the *Variable* class. The full OWL source code of the Bayesian network ontology is shown in Appendix A.

```
{...}
<owl:Class rdf:ID="ChanceVariable">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Variables representing random events. A Chance Variable
   is composed by a label and a set of states representing the
   random events.
  </rdfs:comment>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="hasState"/>
    </owl:onProperty>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
     1
    </owl:minCardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Variable"/>
  </rdfs:subClassOf>
</owl:Class>
{...}
```

Figure 3.8: Snippet of the OWL source code of the Bayesian network ontology.

Since the Bayesian network ontology is a reengineering and extension of a previous work developed in the context of our research group, it was selected as implementation language the OWL. This technology is currently a W3C standard, an additional reason to adopt it.

3.4 Final Considerations

The presented ontology is the result of an effort to remodel and to extend the concepts defined by a previous research (SANTOS, 2006; SANTOS; BOFF; VICARI, 2006) performed by members of the Artificial Intelligence Group of UFRGS. This previous work defined an ontology that covers partially the concepts necessary to represent Bayesian networks. The present proposal formalized the Bayesian network knowledge representation, taking into account its structure, its semantics, and possible future improvements to allow its broader utilization.

It is worth to stress that the ontology itself is an independent contribution of this research. The ontology can be used to represent uncertain knowledge in several contexts (i.e. Semantic Web, content language for communication of agents, and interchange format for Bayesian network tools and applications). The ontology can also be improved or modified to cover different graphical representations, such other probabilistic network dialects, and then to support a greater number of applications. In example, the undirected graph and junction tree extensions, exhibited in the Section 3.2.4, can be used for interoperability in a society of agents responsible for a distributed processing of a Bayesian belief updating.

During the ontology development, it was realized that the arcs among variables can be replaced by a property of the parent chance variables representing causal relations.

Their inverse relation is the consequence, which correspond to a property in the child chance variable indicating the parent nodes. Supposing only Bayesian networks will be specified, the node and the variable concepts can be merged in only one concept, which will maintain the structure of a chance variable.

The semantic restrictions of Bayesian networks were not fully coded as OWL constraints, since some of them are not supported by the language. One example is the verification of cycles in a graph corresponding to a Bayesian network. Such verification should be performed as the ontology is used - as individuals are updated in the knowledge base - but there are no means to specify such restrictions directly on the OWL code.

4 INTEGRATING BDI MODEL AND BAYESIAN NETWORKS

This Chapter presents an integration of BDI agent model and Bayesian networks. The synergistic effects resulting from the integration of these Artificial Intelligence technologies include the capability of dealing with uncertain information in a BDI architecture and the improvement of the agent's cognitive processes.

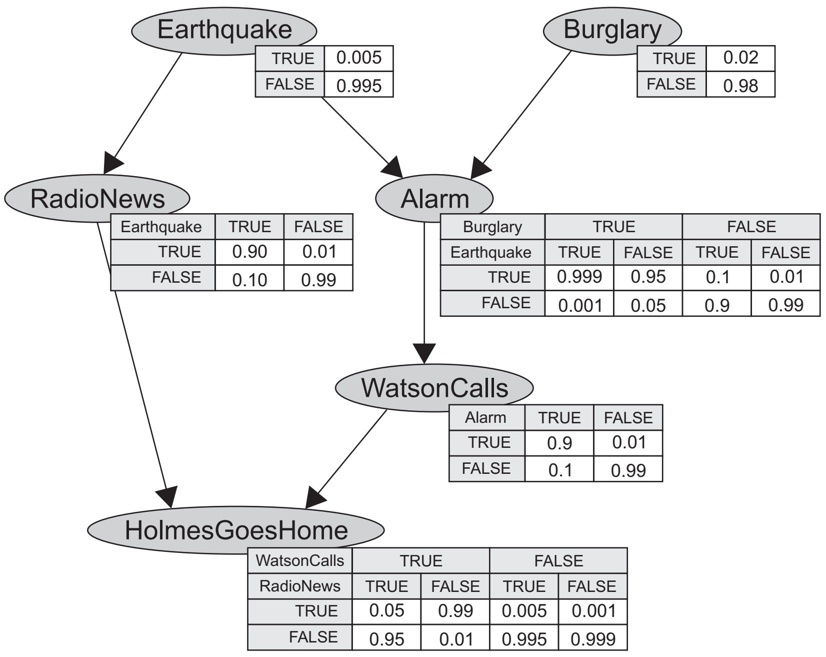
Each node in a Bayesian network corresponds to exactly one random variable which has a finite set of mutually exclusive states. We claim that a BDI agent, whose beliefs are represented through Bayesian networks, believes that a state of a chance variable has a probability of occurrence given a set of conditions imposed by the parent variables. Assuming that desires correspond to states of affairs that an agent wishes to bring about, our model represents this mental state through states of chance variables that the agent desires to observe. Intentions are also represented in this way, since they are desires that an agent has committed to achieve.

Assuming that agent's beliefs are represented through Bayesian networks, the belief updating process correspond to a probabilistic inference. Evidences perceived by the agents play an important role in that process, since up to date beliefs express the current state of world. Thus, agents can recognize circumstances where desires are considered feasible and intentions are considered successful. Up to date beliefs provide support to the deliberative process, responsible for deciding which states of affairs the agent will intend to achieve. In order to improve that process, it takes into account the quantitative and the qualitative aspects of the Bayesian networks to detect incompatible desires and to decide between competing ones.

In cognitive multiagent systems, inhabited by a relative small number of knowledge intense agents, the interoperability is fundamental to enable the information exchange among agents. As presented in the previous Chapter, this work adopts an ontology-based approach to specify the Bayesian network concepts. Agents that share that ontology are able to interoperate that representation of uncertain knowledge. Since the BDI mental states are represented through Bayesian networks and the Bayesian networks' domain is specified in an ontology, the mental states can also be viewed as ontology individuals.

Throughout this Chapter it will be used the *Burglary or Earthquake* Bayesian network (PEARL, 1988) to demonstrate the integration of the BDI model and the Bayesian networks. In this very known example, Mr. Holmes is working in his office when he receives a phone call from his neighbor Dr. Watson, who tells Mr. Holmes that his alarm has gone off. Convinced that a burglar has broken into his house, Mr. Holmes rushes to his car and goes home. On his way home, he listens to the radio, and in the news it is reported that there has been a small earthquake in the area. Knowing that earthquakes have a tendency to make burglar alarms go off, he returns to his work.

The Figure 4.1 illustrates the *Burglary or Earthquake* Bayesian network example. The *HolmesGoesHome* variable, which represents Mr. Holmes going home, does not exist in the original construction of this network. It was added in order to represent a possible world state to be achieved by Mr. Holmes. This example is developed using only Boolean variables, however the proposed agent architecture is capable of dealing with all types of discrete chance variables.



Earthquake	
TRUE	0.005
FALSE	0.995

Burglary	
TRUE	0.02
FALSE	0.98

Earthquake	TRUE	FALSE
TRUE	0.90	0.01
FALSE	0.10	0.99

Burglary	TRUE		FALSE	
Earthquake	TRUE	FALSE	TRUE	FALSE
TRUE	0.999	0.95	0.1	0.01
FALSE	0.001	0.05	0.9	0.99

Alarm	TRUE	FALSE
TRUE	0.9	0.01
FALSE	0.1	0.99

WatsonCalls	TRUE		FALSE	
RadioNews	TRUE	FALSE	TRUE	FALSE
TRUE	0.05	0.99	0.005	0.001
FALSE	0.95	0.01	0.995	0.999

Figure 4.1: *Burglary or Earthquake* Bayesian network.

This Chapter is organized as follows: the Section 4.1 concerns with the architecture's mental states; the cognitive processes are detailed in the Section 4.2; the Section 4.3 addresses the interoperability of Bayesian cognitive agents implemented in compliance with the present architecture; finally, the Section 4.4 presents the final considerations.

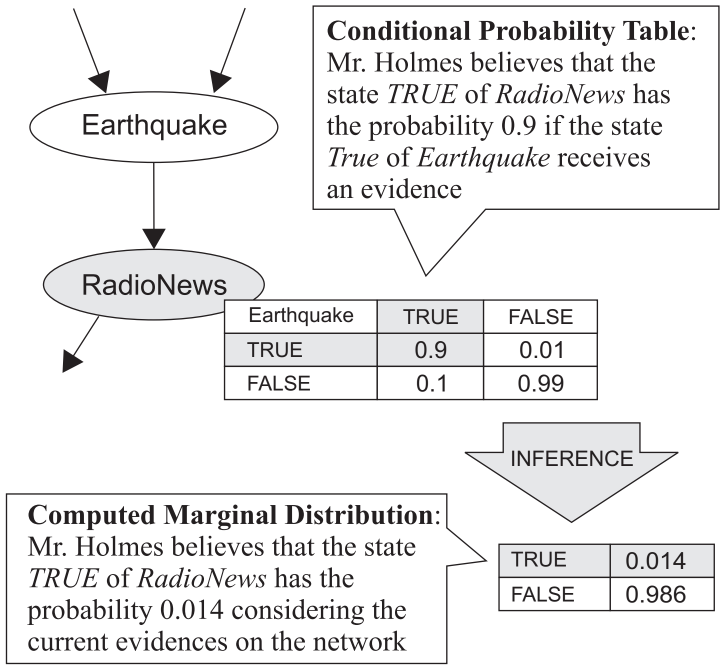
4.1 Representing Mental States through Chance Variables

The proposed BDI architecture represents its mental states through discrete Bayesian networks, more accurately, through an ontology that models the Bayesian network domain. The ontology-based approach is closely related with interoperability issues, later discussed in detail.

Before presenting the representation of mental states, it is worth to stress two distinctions made by Searle (SEARLE, 1983) that are relevant to this Chapter. The former distinguishes *Intentional state*, a state of mind wherein the mind has some object (Intentional content) towards which it is directed, from *Intentional content*, also called propositional content, which determines conditions of satisfaction for the Intentional state. In example, consider a situation where *agent Smith intends to find Neo*. We can claim that the *Intentional state* is an *intention* and the *Intentional content* corresponds to *find Neo*. The last distinction is between *Intentionality*, a property of many mental states by which they are directed at something (abouteness), and *intention*, a mental state whose propositional content represents a state of affairs that an agent has committed to achieve.

4.1.1 Beliefs

Beliefs are informational mental states that express the world's state. In the context of this work, a BDI agent believes that a state has conditional probabilities. In other words, the Intentional content of a belief is composed by a chance variable, a state of that variable, a probability of occurrence of that state, and the conditions (states of parent variables) under which the probability is estimated. Beliefs of prior chance variables are exceptions, since they are unconditional. Such belief abstraction is related to the probabilities specified by the expert (developer) in the conditional probability tables. By performing a Bayesian inference it is obtained the probabilities of each state considering perceived evidences. Thus, agents also believe that states have a computed probability taking into account the current situation of the network. Summarizing, for each variable state we have conditional probabilities defined by the domain expert, and the probability resulting from the Bayesian inference process. Both cases are depicted in the Figure 4.2.


Conditional Probability Table: Mr. Holmes believes that the state *TRUE* of *RadioNews* has the probability 0.9 if the state *True* of *Earthquake* receives an evidence
Earthquake
RadioNews
INFERENCE
Computed Marginal Distribution: Mr. Holmes believes that the state *TRUE* of *RadioNews* has the probability 0.014 considering the current evidences on the network

Earthquake	TRUE	FALSE
TRUE	0.9	0.01
FALSE	0.1	0.99

TRUE	0.014
FALSE	0.986

Figure 4.2: Representation of beliefs.

The Figure 4.2 shows the *Earthquake* and *RadioNews* variables of the Mr. Holmes' beliefs. The remaining variables were omitted. Both tables belong to the *RadioNews* node. The conditional probability table contains the agent's beliefs about the probability of occurrence of states under the observation of conditions (parent nodes' states). One could argue that these beliefs are expectations, since they relate to beliefs in the future (i.e. Mr. Holmes believes that the state *TRUE* of *RadioNews* has the probability *0.9* of occurrence if the state *TRUE* of *Earthquake* gets observed). The probability table of the computed marginal distribution, resulting from the Bayesian inference, expresses the agent's beliefs about the current state of the world (i.e. currently, Mr. Holmes believes that the state *TRUE* has the probability *0.986* of occurrence).

Taking into account that beliefs may change over the time, it is important to specify which Bayesian network elements are mutable. We assume that only the *computed marginal distribution values* (see Figure 3.3 in the Chapter 3) are vulnerable to those changes in order to reflect the evolution of the agent's environment. The remaining elements, such as conditional probability tables, causal relations, variables and states, are immutable. Of course, these restrictions are imposed by the architecture, and to remove them demand the employment of learning techniques (i.e. Expectation Maximization algorithms for parameter and topology learning in Bayesian networks (LUNA, 2004; COWELL et al., 1999)) beyond the probabilistic inference.

Beliefs are updated to express the state of the agent's environment. In the present agent architecture, the changes take place when a probabilistic inference (see Section 4.2.1) is performed based on evidences perceived by the agent. After each probabilistic inference, a new network situation (*BayesianSituation* class individual) arises. The Figure 4.3 illustrates an example where an agent perceives an evidence and updates its beliefs according to the perception.

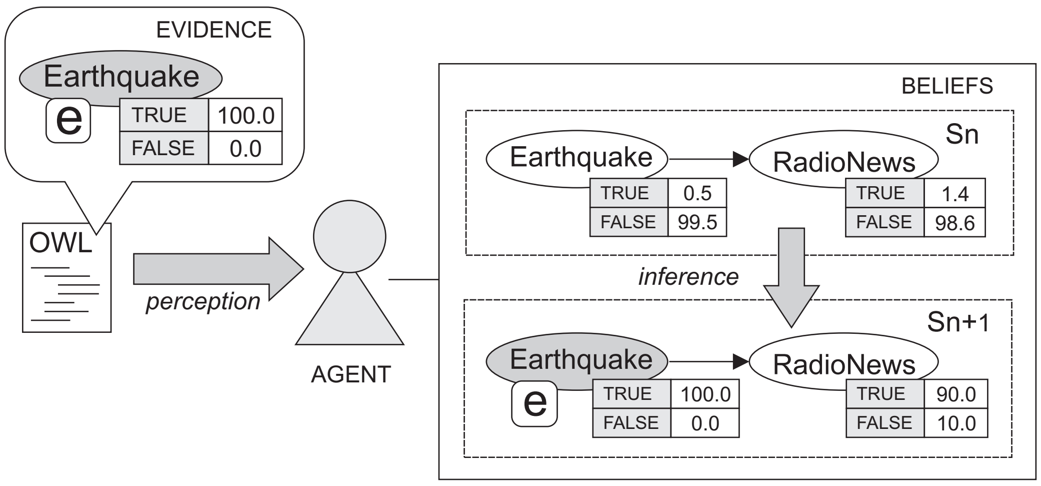


Figure 4.3: An agent perceiving an evidence and updating its beliefs (inferring) according to the perception.

The perception is represented by the gray arrow on the left side of the agent. The perceived evidence, coded in OWL language, indicates that the state *TRUE* of the *Earthquake* variable has been diagnosed with certainty. The Figure 4.3 shows only two variables, *Earthquake* and *RadioNews*, and the last two situations of the agent's network. The first, named *Sn*, represents the current situation before the execution of the inference process that considers the perceived evidence in the variable *Earthquake*. The second is the current situation, called *Sn+1*, resulting from the inference process. The inference is depicted in the figure by the gray arrow from the situation *Sn* to the situation *Sn+1*. Following the information flow: the agent perceives the evidence, and since the state of *Earthquake* is known, it is necessary to perform an inference to recalculate the probabilities associated with the variable *RadioNews*. After the inference execution a new situation *Sn+1* is generated from the situation *Sn*, considering the evidence in the *Earthquake* variable.

A snippet of the OWL source code corresponding to the evidence perceived by the agent is illustrated in the Figure 4.4. It begins by specifying three *Label* individuals. The former represents the variable's name (*Earthquake*) and the remaining the labels of the variable's states (*TRUE* and *FALSE*). After them, it is specified a *PriorChanceVariable* individual, followed by a *ChanceNode* individual. Finally, the *Evidence* individual has three properties that point respectively state's label, its probability, and the node that the state belongs.

The expert responsible for specifying the agent's beliefs have to construct the Bayesian networks in conformance with this knowledge representation, where the causal relations

```
<Label rdf:ID="Label_1">
   <name rdf:datatype="{...}#string">Earthquake</name>
</Label>
<Label rdf:ID="Label_2">
   <name rdf:datatype="{...}#string">TRUE</name>
</Label>
<Label rdf:ID="Label_3">
   <name rdf:datatype="{...}#string">FALSE</name>
</Label>
<PriorChanceVariable rdf:ID="PriorChanceVariable_1">
   <hasLabel rdf:resource="#Label_1"/>
   <hasState>
      {...}
   </hasState>
   <hasMarginalDistribution>
      {...}
   </hasMarginalDistribution>
</PriorChanceVariable>
<ChanceNode rdf:ID="ChanceNode_1">
   <hasChanceVariable
rdf:resource="#PriorChanceVariable_1"/>
   <hasLabel rdf:resource="#Label_1"/>
</ChanceNode>
<Evidence rdf:ID="Evidence_1">
   <hasLabel rdf:resource="#Label_2"/>
   <probability rdf:datatype="{...}#float">1.0</probability>
   <hasNode rdf:resource="#ChanceNode_1"/>
</Evidence>
```

Figure 4.4: Snippet of the OWL code of an evidence.

and the initial conditional probabilities should be as accurate as possible in order to express the domain's reality. This modeling task is fundamental because in some situations the agent will not have evidences about all states it needs, and the agent's decisions will be taken based in the conditional probability tables initially defined. Before perceiving evidences, an agent has only unverified beliefs meaning that it believes that states of world can occur with a conditional probability.

Concerning beliefs, it is worth to stress some points. A variable *Earthquake* in an agent's network has the same meaning of a variable *Earthquake* in a network of other agents. This design decision was taken since the focus of this research is the integration of the BDI model and the Bayesian networks. However, the structure necessary to add semantics to the chance variables is provided. The *ChanceNode*, *ChanceVariable* and *State* classes have a property named *hasLabel*, which may indicate individuals for adding semantics to the concepts represented by those Bayesian network elements. In example, the *hasLabel* property of a *ChanceVariable* could point to an individual of a class named *Earthquake*, which has two properties: *hasMagnitude* and *hasArea*. The *hasArea* property can be a map of the area affected by the earthquake. The magnitude of an earthquake, measured in a scale (i.e. Richter), can be denoted by a property named *hasMagnitude*. It is clear that the ontological modeling of those concepts could be as detailed as necessary.

It is allowed the existence of Bayesian networks with only one node and without relations. One could argue that they are not networks, but in the context of this cognitive agent architecture they are viewed as beliefs without causal relations.

4.1.2 Desires and Intentions

Desires are Intentional mental states that represent the states of affairs that an agent would, in an ideal world, wish to be brought about, and intentions represent desires that it has committed to achieving (WOOLDRIDGE, 2000).

Since desires are states of affairs that an agent wishes to bring about, this work considers that desires correspond to states of variables. The intentions are also represented in this way, since the only difference between them and the desires is the commitment established by the agent with the intentions. Assuming that desires and intentions relate to particular states of the world (chance variable states) to be achieved, those mental states are considered successful if the agent believes in their Intentional content (the variable states have been observed).

The Figure 4.5 illustrates how the desires are viewed through Bayesian networks. The desire consists of the variable and the desired state. In this case, Mr. Holmes desires the state *TRUE* of the *HolmesGoesHome* variable. The probabilities associated with each state of *HolmesGoesHome* are conditioned by the states of the parent variables. The desire selection, discussed in detail in the deliberation process Section, takes into account the probabilities of those states of parent variables.

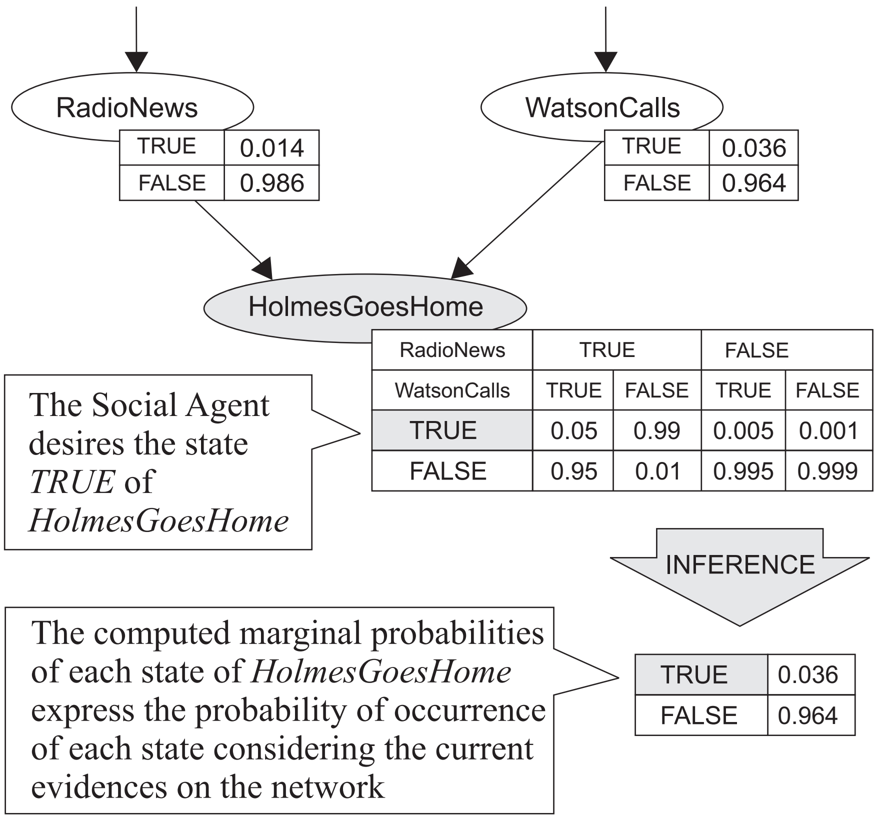

RadioNews
TRUE 0.014
FALSE 0.986
WatsonCalls
TRUE 0.036
FALSE 0.964
HolmesGoesHome
RadioNews TRUE FALSE
WatsonCalls TRUE FALSE TRUE FALSE
TRUE 0.05 0.99 0.005 0.001
FALSE 0.95 0.01 0.995 0.999
The Social Agent desires the state TRUE of HolmesGoesHome
INFERENCE
TRUE 0.036
FALSE 0.964
The computed marginal probabilities of each state of HolmesGoesHome express the probability of occurrence of each state considering the current evidences on the network

Figure 4.5: Representation of desires through Bayesian networks.

In opposition to beliefs, the Intentional state of desires and intentions were explicitly represented because there is not a way to identify these mental states just by checking the networks. Since desires and intentions relates to particular states of the world (chance variable states) to be achieved, it is possible to assert that the Intentional content of desires and intentions are subsets of beliefs. Besides, it is possible to assert that intentions are a subset of desires, assuming that it is possible to intend a state that is desired.

These proactive Intentional states were specified as OWL ontology classes, which are illustrated by the conceptual map in the Figure 4.6. To represent a proactive mental state it was created the *ProActiveMentalState* class, which has two properties: *hasState* and *hasChanceVariable*. These properties indicate the Intentional content of a proactive mental state, that is, its conditions of satisfaction. In this work, the Intentional content of proactive mental states corresponds to a chance variable and one of its states. Two classes are specialized from *ProActiveMentalState*: *Desire* and *Intention*, which do not specify additional properties. This distinction is justified by the difference of meaning between these two mental states: both represent state of affairs to be achieved, but intentions represent the agent's commitment to those states.

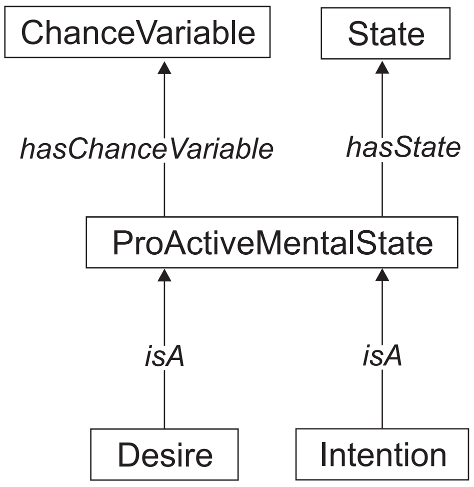


Figure 4.6: Representation of the desires and intentions concepts and relationships.

In variables of prior nodes, the Intentional content of proactive mental states is represented by a *PriorChanceVariable* and a *StateProbability* individual. In non prior node variables, the Intentional content of desires and intentions is represented by a *ConditionalChanceVariable* and a *ConditionalProbability* individual.

The Figure 4.7 illustrates an OWL code snippet an intention. It corresponds to an intention to achieve the state *TRUE* of a chance variable labeled *HolmesGoesHome* (see Figure 4.1). The code begins by specifying two labels (Label individual). Following them, there are a prior chance variable (*PriorChanceVariable* individual) and one of its states (*StateProbability* individual). Finally, it is specified the Intention individual.

Desires of a BDI agent do not need to be compatible, since there is no commitment to achieve them. Consider a situation where an agent desires multiple states of the same chance variable. We know that these states are mutually exclusive, but from a BDI viewpoint it is not wrong to desire them. Based on this assertion, the architecture allows an agent to desire more than just one state, though only one state can be intended.

It is known that intentions should be consistent. Thus, an agent can not entertain options that are not consistent with its intentions. The only restriction explicitly imposed by the Bayesian networks is the mutual exclusiveness of the states, which denies intentions to point to states of the same chance variable. However, in some way, the agents have to know which intentions can not coexist. This issue is addressed by the deliberation process.

```
<Label rdf:ID="Label_4”>
   <name rdf:datatype="{...}#string">HolmesGoesHome</name>
</Label>
<Label rdf:ID="Label_2">
   <name rdf:datatype="{...}#string">TRUE</name>
</Label>
 {...}
<PriorChanceVariable rdf:ID="PriorChanceVariable_2">
   <hasLabel rdf:resource="#Label_4"/>
   <hasState>
      {...}
   </hasState>
   <hasMarginalDistribution>
      {...}
   </hasMarginalDistribution>
</PriorChanceVariable>
<StateProbability rdf:ID="StateProbability_2">
   <hasLabel rdf:resource="#Label_2"/>
   <probability rdf:datatype="{...}#float">1.0</probability>
 </StateProbability>
 <Intention rdf:ID="Intention_1">
   <hasState rdf:resource="#StateProbability_2"/>
   <hasChanceVariable rdf:resource="#PriorChanceVariable_2"/>
  </Intention>
```

Figure 4.7: Snippet of the OWL code of an intention.

4.2 Processes

Once introduced the beliefs, desires and intentions, it is detailed the processes responsible for manipulating these mental states. The flow of information in our agent model begins by perceiving the environment and updating the beliefs. Up to date beliefs, together with desires and intentions, provide the support for the deliberation process, which results in the intentions to be pursued by the agent.

4.2.1 Belief Updating

To keep beliefs up to date is a crucial task to an agent, since in a dynamic world it is necessary to make decisions and execute actions taking into account the state of the environment.

The belief updating process is triggered when agents perceive new evidences. The evidences, perceived as *Evidence* class individuals, have to be analyzed by the agent in order to associate them with the chance variables. Concerning analysis, the simplest one consists in comparing the variable pointed by the evidence with the agent’s variables. It is assumed that the *hasLabel* property of the *ChanceVariable* class can indicate other individuals except the *Label* ones, what gave rise to a better description of the concept modeled by the chance variable.

In the Figure 4.3 the agent perceives an evidence indicating that an *Earthquake* has occurred. Comparing this evidence with its beliefs, the agent realizes that it has a variable that represents this event. Therefore, the agent performs an inference which updates the chances of its beliefs.

As stated in the Section 3.2.3, a Bayesian network situation consists of a particular configuration that a network assumes given a set of evidences. The current situation of a network is computed considering the most recent evidences and it is the most up to date knowledge the agent has about its environment. A situation is represented by a *BayesianSituation* class individual, and a transition between two situations is represented by a *BayesianSituationTransition* class individual. The evidence (*Evidence* individual) and the current situation (*BayesianSituation* individual) are the Bayesian inference input. As output it has a *BayesianSituation* individual, representing the new situation, and a *BayesianSituationTransition* individual, establishing a link between the new situation and that used as input.

4.2.2 Deliberating Under Uncertainty

In the deliberation process, BDI agents examine which desires are possible, choose between competing desires, and commit to achieving them (WOOLDRIDGE, 2000). In this work, the deliberation process was divided in two stages: the first verifies which desires are possible to be achieved, and the second checks the compatibility among possible desires and intentions.

Possible desires are those considered reachable by the agent. In order to select the possible desires, the agent checks the *availability of action plans* and the *probabilities of the desired states* (degree of belief in the desired state). The action plans are closely related to the construction of the Bayesian networks. By this it means that the action plans are suppose to cause particular variable states, and the combination of action plans can be made to obtain a condition (combination of variable states). The state *TRUE* of the variable *HolmesGoesHome* in the Figure 4.1 is desired by Mr. Holmes. In order to achieve that state, the agent has an available action plans that consists in driving home with his car. The action plans are not represented in the Bayesian networks, only the state of world that they are supposed to cause.

If there is at least one action plan available to achieve the desired state, the second step to decide which desires are possible is the evaluation of the computed marginal probability associated with the desired state. Assuming that the probabilities associated with the states indicate their chance of occurrence, and desires are represented by variable states, it is possible to assert that the probabilities indicate the chance of occurrence of the desired state. In this work, a BDI agent will intend only states of the world that it believes to be possible to achieve. Some applications require *cautious agents*, which act only under high probabilities. Other applications demand *bold agents*, which act also under low probabilities. It means that the threshold used to select the desires is specified in accordance with the expected behavior.

In example, consider that Mr. Holmes goes home when the probability of the state *TRUE* of *HolmesGoesHome* gets higher than *0.5*. By adopting that bold behavior, Mr. Holmes is more susceptible to the intention failure. However, Mr. Holmes will go home in most part of times that a burglar breaks into his house. Now, assume that Mr. Holmes goes home only when the probability of the state *TRUE* of *HolmesGoesHome* gets higher than *0.99*. That very cautious behavior will guarantee that the intention will be created only under a high degree of certainty, but it decreases the chances of reaction to an ongoing burglary.

It is known that BDI agent can entertain incompatible (mutually exclusive) desires, but it is not allowed to commit to them. The incompatibility is detected in two contexts: *local* and *global*. The *local incompatibility* happens between states of the same chance variable. Since they are mutually exclusive, an agent can not intend more than one. The *global incompatibility* happens on the network context. Desires represented by states in different chance variables are incompatible when the achievement of one decreases the probabilities of occurrence of another. The evaluation of the degree of incompatibility (how significant is the decrease of probability?) fits to the threshold issue, previously discussed: in some domain of application a small decrease is not significant and in other domain it is. For a cautious agent, who intends a state only under high probabilities, any decrease of chances can mean an intolerable incompatibility.

To detect *global incompatibilities*, the agent simulates a fake evidence indicating that the desired state has been observed. Once forged the fake evidence, a probabilistic inference is performed. If the resulting network situation exhibits an interference on probabilities of other desires, there is an incompatibility among those mental states.

The second stage begins by checking the compatibility (global and local) of the possible desires (those approved on the previous stage) with the agent's intentions. In this work, the desires considered incompatible with the agent's intentions are dropped. In some circumstances to drop an intention to select a desire can be advantageous, however this research does not address issues such as commitment strategies and intention reconsideration.

Desires compatible with the agent's current intentions compete between themselves to become intention. Differently from checking compatibility among desires and intentions, now the agent has to employ a criterion for deciding between competing desires. The main criterion employed by this approach consists in checking the probability associated with the desired state. It is known that this probability does not mean the chance of success of a plan, but the chance of occurrence of that state. Since our agent desires states of affairs that it believes that are possible, it will decide in favor of the desire with the highest probability.

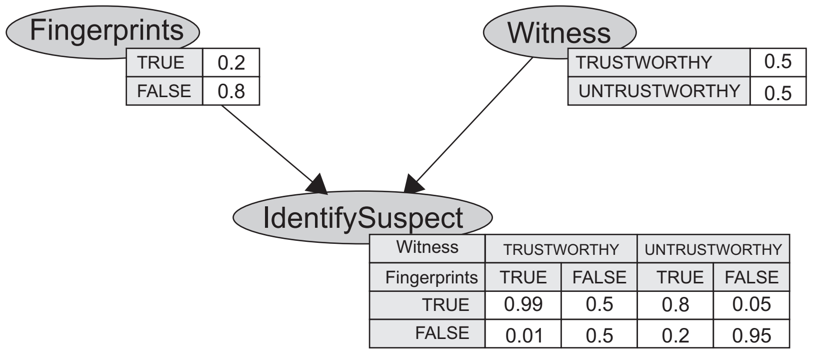
It is important to point out that the strategy used to set the threshold has to take into account the quantitative aspects of agent's beliefs. The maximum probability value that can be inferred in the state *TRUE* of the variable *HolmesGoesHome* is *0.99*, if the state *FALSE* of the variable *RadioNews* and the state *TRUE* of the variable *WatsonCalls* get observed. To assume a value higher than *0.99*, the state *TRUE* of *HolmesGoesHome* has to receive an evidence. Supposing that a very cautious agent set the threshold value to *0.999*, the desire *TRUE* of *HolmesGoesHome* will never be selected as a possible one.

The desires selected by the deliberation process become intentions, leading the agents to action through their plans. Intentions are considered successful if the intended state receives an evidence indicating that it has been observed with certainty. Otherwise, the agent assumes that the intention has failed.

In the *Earthquake or Burglary* example, the agent will wait until the world becomes favorable to its desire. An evidence on *WatsonCalls* is enough to increase the chance of *HolmesGoesHome* from *0.036* to *0.967*. Supposing a threshold of *0.9*, Holmes will immediately go home. However, in some cases, agents have available action plans to change the world state. By this it means that agent can affect the parent variables to produce a situation where desires become possible. Those action plans are based on the relation among variables. Although that kind of relation does not necessarily imply in causality (JENSEN, 2001), the parent variables have an influence on the probabilities of the conditioned variables that did not receive evidences.

Until now it was discussed only action plans in the context of the desired state. This kind of plan is named *local plan* because it is limited to the variable context. Of course, a successful local plan execution will collaterally affect states of other unobserved variables. The local action plan is executed locally in the context of the variable, but its effects can be noticed by other variables. By acting in the parent variables, the agents can cause a state of world where their desires are feasible. This kind of plan, that intends to change the states of parent variables, is named *global plan*. Global plans are assembled at runtime, and they can be extended recursively to the parent variables of parent variables.

Before explaining how global plans are assembled, it is presented the *Identify Suspect* example. In this example, Mr. Holmes acts proactively to produce a world state where the desire to identify the suspect becomes possible to achieve. The Figure 4.8 illustrates the Holmes' beliefs. The Bayesian network with three variables: *Witness*, *Fingerprints* and *IdentifySuspect*. The *Witness* variable models someone who saw the crime and accused the suspect. Witness variable can assume two states: *trustworthy* or *untrustworthy*. The *Fingerprints* variable informs if there are fingerprints of the suspect in the gun used in the crime. The *TRUE* state means that there are fingerprints of the suspect. The *IdentifySuspect* chance variable has two states: *TRUE* and *FALSE*. If the variable assumes the state *TRUE*, the suspect has been identified. Holmes desires to achieve the state *TRUE* of the node *IdentifySuspect*, and identify the author of the crime. Differently from the previous example, now Mr. Holmes has actions available make the desire feasible. The action plans, presented now in a high level of abstraction, are to analyze the fingerprints and to interrogate the witness. After executing those action plans, it is expected evidences indicating the chances of *Witness* and *Fingerprints*.



TRUE	0.2
FALSE	0.8

TRUSTWORTHY	0.5
UNTRUSTWORTHY	0.5

Witness	TRUSTWORTHY		UNTRUSTWORTHY	
Fingerprints	TRUE	FALSE	TRUE	FALSE
TRUE	0.99	0.5	0.8	0.05
FALSE	0.01	0.5	0.2	0.95

Figure 4.8: Identify Suspect example.

The requirement for assembling global plans is the availability of actions to change particular states of the desire's parent variables. The plan construction begins by checking the conditional probabilities of the desired variable. In example, to increase the probability of the state *TRUE* of the variable *IdentifySuspect*, Mr. Holmes checks the conditional probabilities of that state, and realizes that if the states trustworthy of *Witness* and *TRUE* of *Fingerprints* get observed, the probability of the state *TRUE* of *IdentifySuspect* goes to *0.99*. The second entry has probability *0.5* and the conditions are composed by the states *FALSE* of *Fingerprints* and *trustworthy* of *Witness*. All entries are checked, and the available plans are stored within the desire.

To update the beliefs before deliberating is crucial, since evidences can dismiss the necessity of executing actions. Supposing that the state *trustworthy* of *Witness* receives an evidence, an available action for *TRUE* of *Fingerprints* would increase the chances of the state *TRUE* of the variable *IdentifySuspect*.

Global plans have to be checked for incompatibilities in the same way as local plans. The second stage of the deliberation process also applies to the intended states of the parent variables. By this it means that intended states of global plans have to present local compatibility, in the variable context, and global compatibility, in the Bayesian network context. Thus, for each intended state of parent variables, the agent simulates a fake evidence and performs a Bayesian inference. Global plans considered incompatible are dismissed. Desires are considered incompatible if all of its plans are considered incompatible.

Agents can be programmed to take decisions under uncertainty, where there are not evidences in all parent nodes, just inferred probabilities. Suppose that the *Fingerprints* variable, in the example of the Figure 4.8, specifies *0.99* as chance of the state *TRUE* instead of *0.2*. That high probability would be specified in a circumstance where the suspect was viewed with a gun in his hands in the crime scene.

Again, it is worthy to stress that global plans intend to achieve states of world where desires become feasible. Given that affirmation, the deliberation process could give priority to those desires which have only local plans, since the state of world is already favorable to them. In circumstances where the agent has not actions, it has to wait until the environment becomes favorable, or interact cooperatively with other agents.

4.3 Interoperability Example

An example of interoperability between two BDI agents, Watson and Holmes, is illustrated in the Figure 4.9. The beliefs of the agent Watson are represented through a Bayesian network with two nodes: *Alarm* and *WatsonCalls*. If it perceives the Holmes' alarm has gone off, it calls Holmes. The beliefs of Holmes, presented in the Figure 4.1, are partially exhibited in the Figure 4.9. Since this Section focuses on the integration of BDI model and Bayesian networks, it is considered that variables in different networks, but with the same label, have the same semantics. In other words, the node *Alarm* in the Holmes' beliefs has the same meaning of the node *Alarm* in the Watson's beliefs. As previously stated, ontology classes representing these concepts can be developed and associated with the chance variables and states.

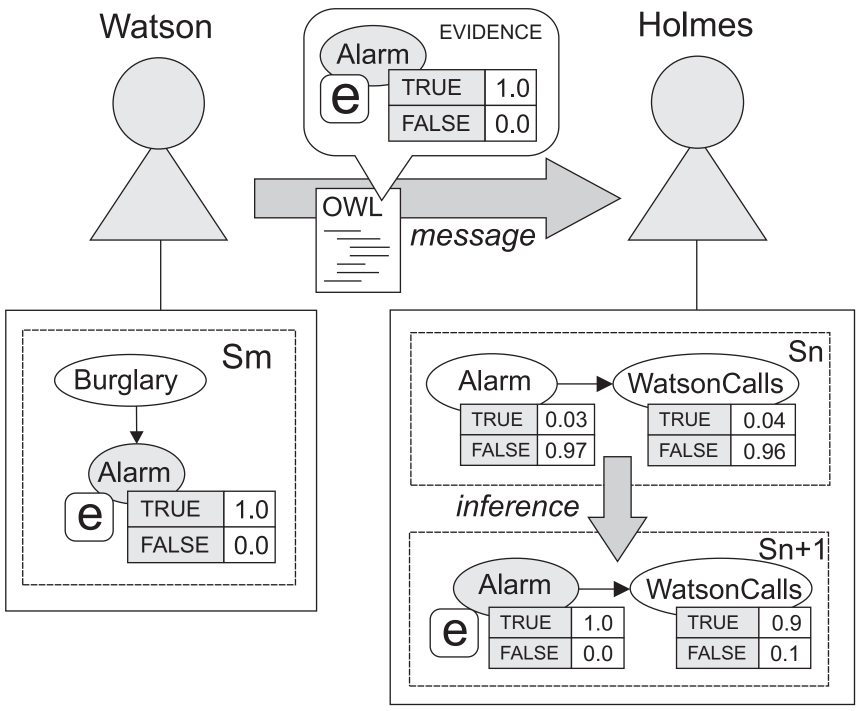


Figure 4.9: Interoperability example.

In the Figure 4.9 is depicted only the current situation *Sm* of Watson's beliefs. In the situation *Sm*, the node *Alarm* has an evidence that indicates the occurrence of the state *TRUE*. The Bayesian network of Holmes, as stated above, is partially represented. It is shown the last two situations of Holmes' beliefs. The first, named *Sn*, represents the current situation before the execution of the Bayesian inference process that considered the received evidence in the node *WatsonCalls*. The second is the actual situation, called *Sn+1*, resulting from the inference process. The inference is illustrated in the figure by a gray arrow from the situation *Sn* to the situation *Sn+1*.

A message exchanged among Watson and Holmes is represented by the gray arrow between them. The message, written in OWL, contains an evidence associated with the node *WatsonCalls*. The Figure 4.10 presents a snippet of the OWL source code corresponding to the message content.

```
<Label rdf:ID="Label_1">
   <name rdf:datatype="{...}#string">Alarm</name>
</Label>
<Label rdf:ID="Label_2">
   <name rdf:datatype="{...}#string">TRUE</name>
</Label>
<Label rdf:ID="Label_3">
   <name rdf:datatype="{...}#string">FALSE</name>
</Label>
<ConditionalChanceVariable rdf:ID="ChanceVariable_1">
   <hasLabel rdf:resource="#Label_1"/>
   <hasState>
      {...}
   </hasState>
   <hasMarginalDistribution>
      {...}
   </hasMarginalDistribution>
</ConditionalChanceVariable>
<ChanceNode rdf:ID="ChanceNode_1">
   <hasChanceVariable rdf:resource="#ChanceVariable_1"/>
   <hasLabel rdf:resource="#Label_1"/>
</ChanceNode>
<Evidence rdf:ID="Evidence_1">
   <hasLabel rdf:resource="#Label_2"/>
   <probability rdf:datatype="{...}#float">1.0</probability>
   <hasNode rdf:resource="#ChanceNode_1"/>
</Evidence>
```

Figure 4.10: Snippet of the OWL source code corresponding to a evidence.

Following the flow of information, Watson sends to Holmes a message containing the evidence associated with the node *WatsonCalls*. Upon receiving the OWL message, Holmes analyzes the message content and realizes that it is informing an evidence that *WatsonCalls* is *TRUE*. It means that Watson perceived that the alarm is gone off and called to Holmes to notice a possible burglary. Since the state of *WatsonCalls* is known, it is necessary to perform the inference to recalculate the probabilities of the network. The inference generates a new situation *Sn+1* from the situation *Sn* considering the evidence from the node *WatsonCalls*. Changes on beliefs may give rise to intentions through deliberation process, in this case the intention of achieving the state *TRUE* of *HolmesGoesHome*.

4.4 Final Considerations

In this Chapter it was presented an integration of the BDI agent model and Bayesian networks, which gave rise to an agent model capable of reasoning under uncertainty. It was proposed a straightforward way to abstract beliefs, desires and intentions through Bayesian networks, making possible to represent probabilistic information within those mental states. To keep beliefs up to date, it was employed the Bayesian inference process. That evidence-based reasoning updates the conditional probabilities taking into account the relations among beliefs.

The Bayesian networks were codified in an ontology language (Chapter 3), allowing the agents to interoperate with other agents that share those ontology concepts. The understanding about the Bayesian networks' domain provided by the ontology, more specifically by conceptual modeling task, facilitated the integration of the BDI model and Bayesian networks, since concepts and relationships were clearly defined.

The deliberative process is employed to cover the goal-directed behavior of the agent. This process is improved by the Bayesian networks, allowing the estimative of probabilities of state occurrence, the verification of compatibility between competing desires and the assembly of global plans. The means-end reasoning of our agents is embedded inside the deliberative process. By this it means that they decide how to achieve a desire before committing to it.

Distinct behaviors can be obtained by tuning the way that the agent evaluates the probabilities in the deliberation process. Cautious agents desire only states that they believe that have high chances of occurrence. On the other hand, audacious agents do not require high probabilities to desire a state. We claim that this tuning is domain specific, since different applications call for different agent behaviors.

The Bayesian networks, through their causal relations, provide a support that enables the agent to act in order to increase the chances of occurrence of desired states. Global action plans are composed in a dynamic way, which gives flexibility to the model since the agent can construct plans considering the current state of world. The global plan construction can be considered a proactive attempt to change the environment. BDI models such as AgentSpeak(L) and Procedural Reasoning System perform a plan only when a context (condition) is established. The inexistence of causal relations among beliefs in those models does not allow the causal reasoning. BayesJason represents beliefs through Bayesian networks, but the deliberative process does not explore aspects of that knowledge representation to promote improvements on the BDI model.

Finally, there is not a unique implementation of the proposed integration. Several different model can be obtained by taking into account different aspects. In example, a particular agent implementation can exhibit a cautious behavior, performing local plans under high probabilities, and assembling global plans only if it has actions to achieve all intended states of parent nodes.

5 CASE STUDY

This Chapter presents a case study, which consists in applying the agent architecture, presented in the previous chapter, in the Social Agent. By this, it is intended to discuss the pragmatic aspects of the proposed integration and present how this dissertation fits in the AIG research.

5.1 Context

The PortEdu (NAKAYAMA; VICARI; COELHO, 2005) is a portal that provides access to educational content and systems. That portal is a FIPA-compliant (FIPA: Foundation For Intelligent Physical Agents, 2006) multiagent system, designed to support agent-based learning environments, such as AMPLIA (VICARI et al., 2003). Among PortEdu's features are the Information Retrieval Agent (NAKAYAMA; VICARI; COELHO, 2005), the User Profile Agent (ALMEIDA, 2004) and the Social Agent (BOFF; SANTOS; VICARI, 2006).

The AMPLIA is an intelligent multiagent learning environment, designed to support training of diagnostic reasoning and modeling of domains with complex and uncertain knowledge, which focuses on the medical area. The AMPLIA's functionalities are also provided by an agent society, which includes the Learner Agent and the Mediator Agent. The Learner Agent represents the student beliefs in a specific domain and the confidence degree this learner has on its tasks. The Mediator Agent has knowledge about the domain, in this case, the medical one. It allows the Mediator Agent to evaluate the student actions outcome.

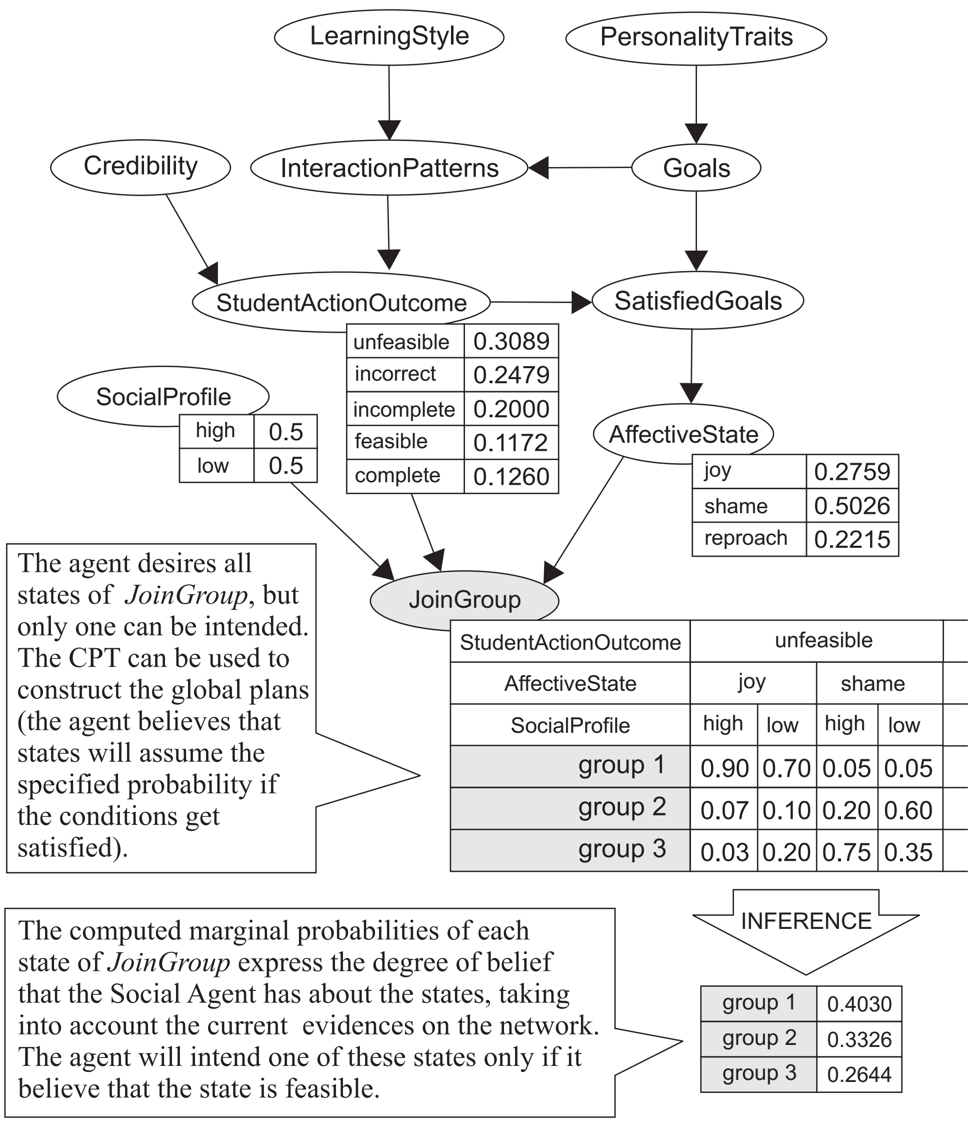
This case study is specifically concerned with the Social Agent, which is currently being developed by Elisa Boff in an ongoing PhD thesis of the PPGC/UFRGS. The main goal of the Social Agent is to improve student's learning by stimulating his interaction with other students, tutors and professors. The interaction is stimulated by recommending the students to join workgroups to provide and receive help from other students.

To reach its purposes, the Social Agent interacts with other agents following the FIPA specifications, which are considered the current standard for interoperability among heterogeneous agents and, since 2005, are sponsored by IEEE. However, the FIPA standard solves the interoperability issue just at the syntactic level. The semantics of the messages' content is addressed by the ontology presented in the Chapter 3.

5.2 Social Agent

In the previous implementation of the Social Agent (BOFF; SANTOS; VICARI, 2006) it was employed Influence Diagrams (COWELL et al., 1999) as knowledge representation. This approach exhibited gaps and limitations. Influence Diagrams do not represent states of the world to be achieved, just decisions that may lead the agent to those states. Consequently, it is not possible to represent incompatible states of affairs that an agent wishes to bring about. The decisions taken in the Influence Diagrams are sequential and they are evaluated through utility functions. They do not take into account the relations among variables, which together with the quantitative aspects can give rise to a more sophisticated reasoning.

The Figure 5.1 depicts part of the Social Agent's mental states (Bayesian network) about a student. There is one Bayesian network instance for each student that the agent intends to help. The integration with the BDI model eliminated the necessity of using decision and utility nodes. This is the reason for representing beliefs through Bayesian networks, instead of Influence Diagrams.



StudentActionOutcome	
unfeasible	0.3089
incorrect	0.2479
incomplete	0.2000
feasible	0.1172
complete	0.1260

SocialProfile	
high	0.5
low	0.5

AffectiveState	
joy	0.2759
shame	0.5026
reproach	0.2215

The agent desires all states of *JoinGroup*, but only one can be intended. The CPT can be used to construct the global plans (the agent believes that states will assume the specified probability if the conditions get satisfied).

StudentActionOutcome	unfeasible			
AffectiveState	joy		shame	
SocialProfile	high	low	high	low
group 1	0.90	0.70	0.05	0.05
group 2	0.07	0.10	0.20	0.60
group 3	0.03	0.20	0.75	0.35

The computed marginal probabilities of each state of *JoinGroup* express the degree of belief that the Social Agent has about the states, taking into account the current evidences on the network. The agent will intend one of these states only if it believe that the state is feasible.

group 1	0.4030
group 2	0.3326
group 3	0.2644

Figure 5.1: Social Agent's mental states.

The *SocialProfile* is built during the students' interaction. This node has the states *high* and *low* sociability. The student's *AffectiveState* is composed by three states: *joy*, *reproach* and *shame*. From this variable it is also possible to infer distress, admiration and pride. The variable *StudentActionOutcome* represents information about a problem solution given by the student. In this case study, the evidences for this variable are provided by the educational environments hosted in PortEdu, and the variable can assume the states *incorrect*, *feasible*, *unfeasible*, *incomplete*, and *complete*. The chance variable *JoinGroup* represents the student workgroups that can be suggested to the student by the Social Agent. Each group corresponds to a state. Its conditional probabilities indicate the chances of the student to join that workgroup given the conditions. The remaining variables and the pedagogical aspects are beyond the scope of this case study.

In the Figure 5.1 there are two tables associated with the *JoinGroup* variable. The conditional probability table contains the agent's beliefs about the chance of occurrence of states under the observation of conditions. The Social Agent believes that the state group1 of *JoinGroup* has the probability *0.9* of occurrence if the state *unfeasible* of *StudentActionOutcome*, *high* of *SocialProfile* and *joy* of *AffectiveState* get observed. The probability table of the computed marginal distribution, resulting from the Bayesian inference process, expresses the agent's beliefs about the current state of the world. The Social Agent believes that the state *group1* has the probability *0.4030* of occurrence. The desires are pointed out by the gray elements: the states *group1*, *group2* and *group3*, and the variable *JoinGroup*. The Social Agent desires all states of the JoinGroup variable, despite the fact that only one of them can be intended. The probabilities associated with each state of the *JoinGroup* CPT are conditioned by the states of the parent variables. The Figure 5.2 shows a snippet of the OWL code corresponding to the Social Agent's desire *group1* of *JoinGroup*.

It begins by specifying the four *Label* individuals: *JoinGroup* (1-3), *group1* (4-6), *group2* (7-9) and *group3* (10-12). The *Label* individuals are identified by a string datatype. Following the labels, there is a *ConditionalChanceVariable* individual (14-17), whose *hasLabel* property points to the label *JoinGroup*. The *StateProbability* individual (18-21) is specified after the chance variable. It points to the *group1* label and has the probability value equal *1.0*. Finally, the *Desire* individual (22-25) indicates the chance variable and the state probability. It means that the Social Agent desires to bring about the state *group1* of the variable *JoinGroup*. The probability *1.0* indicates that the event have to occur to make the desire successful.

The Figure 5.3 illustrates the Mediator Agent sending a FIPA-ACL message to Social Agent. The message performative is an inform, the content language is the OWL, and the agreed ontology specifies the Bayesian network domain.

In the message content is the OWL code of an Evidence individual that indicates the observation of the state complete in the node StudentActionOutcome. The reception of this evidence by the Social Agent will trigger the Bayesian inference, generating a new situation in the Bayesian network (beliefs).

Assuming that the Social Agent has three desires that belong to the same variable (*group1*, *group2* and *group3* of *JoinGroup*), the deliberation process can choose only one to become intention. This process begins by checking the availability of actions to bring about each desired state. The Social Agent action plan consists in suggesting a particular workgroup to the student. Since the agent has action plans to achieve the three desires, the second stage (evaluation of the probabilities) is performed. It begins by checking the probability of the state group1. In the Figure 5.1, that probability corresponds to *0.4030*.

```
<Label rdf:ID="Label_4”>
  <name rdf:datatype="{...}#string">JoinGroup</name>
</Label>
<Label rdf:ID="Label_1">
   <name rdf:datatype="{...}#string">group1</name>
</Label>
<Label rdf:ID="Label_2">
   <name rdf:datatype="{...}#string">group2</name>
</Label>
<Label rdf:ID="Label_3">
   <name rdf:datatype="{...}#string">group3</name>
</Label>
 {...}
<ConditionalChanceVariable rdf:ID="PriorChanceVariable_1">
   <hasLabel rdf:resource="#Label_4"/>
     {...}
</ConditionalChanceVariable>
<StateProbability rdf:ID="StateProbability_1">
  <hasLabel rdf:resource="#Label_1"/>
   <probability rdf:datatype="{...}#float">1.0</probability>
 </StateProbability>
 <Desire rdf:ID="Desire_1">
   <hasState rdf:resource="#StateProbability_1"/>
   <hasChanceVariable rdf:resource="#PriorChanceVariable_1"/>
  </Desire>
 {...}
```

Figure 5.2: Snippet of the OWL source code of a desire of the Social Agent.

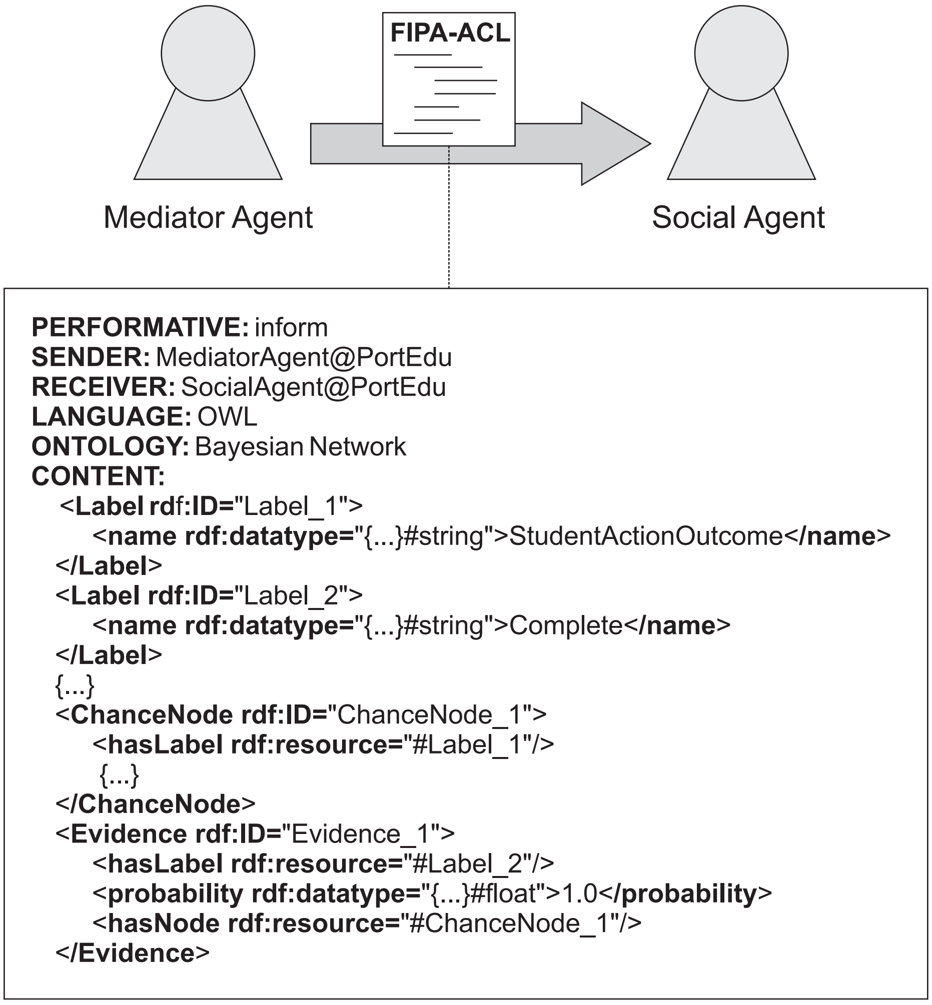
If the threshold value is lower than this value, the Social Agent will consider the *group1* desire unfeasible. The remaining desires have chances lower than *0.4030*, thus they will also be considered unfeasible.

Desires that have an action plan available, but have been considered unfeasible by the probability checking stage, will be forwarded to the global plan verification. Global plans, as stated in the Chapter 4, intend to achieve a world state where desires become feasible. In this case study, the Social Agent has not available action plans to change the states of the parent variables of *JoinGroup*. Therefore, the agent will have to wait until the world becomes favorable.

To update the beliefs before deliberating is crucial, since evidences can dismiss the necessity of executing action plans. Supposing that the states *joy* of *AffectiveState* and *unfeasible* of *StudentActionOutcome* receive evidences, an available action for *high* of *SocialProfile* would turn the state *group1* feasible.

5.3 Final Considerations

It was concluded that the Bayesian network ontology can be integrated with the FIPA standards, more specifically with the FIPA-ACL. The adoption of OWL as a content language for FIPA-ACL messages handles the issue of a common knowledge language. The ontology aggregates meaning to the message content. The utilization of the OWL and the specification of the ontology to contextualize the content, allow the expression of knowl-



```
PERFORMATIVE: inform
SENDER: MediatorAgent@PortEdu
RECEIVER: SocialAgent@PortEdu
LANGUAGE: OWL
ONTOLOGY: Bayesian Network
CONTENT:
  <Label rdf:ID="Label_1">
     <name rdf:datatype="{...}#string">StudentActionOutcome</name>
  </Label>
  <Label rdf:ID="Label_2">
     <name rdf:datatype="{...}#string">Complete</name>
  </Label>
  {...}
  <ChanceNode rdf:ID="ChanceNode_1">
     <hasLabel rdf:resource="#Label_1"/>
      {...}
  </ChanceNode>
  <Evidence rdf:ID="Evidence_1">
     <hasLabel rdf:resource="#Label_2"/>
     <probability rdf:datatype="{...}#float">1.0</probability>
     <hasNode rdf:resource="#ChanceNode_1"/>
  </Evidence>
```

Figure 5.3: Interoperability among Social Agent and Learner Agent.

edge in an open and explicit way.

By applying the BDI architecture to the Social Agent, it was constructed a basis where cooperative tasks can be performed. Since PortEdu is a multiagent environment, inhabited by knowledge intense agents, the possibility of exchange evidences enables those agents to update their beliefs and to achieve a more reasonable behavior under uncertainty. The BDI Social Agent, differently from the previous implementation, has an explicit representation of the states of affairs to be achieved. By representing beliefs about the states to be achieved, the agent can recognize when a plan has been successful. A deliberation mechanism, instead of decision and utility variables, enabled the Social Agent to check for incompatible desires, and to select desires based on feasibility and action availability.

6 CONCLUSION AND FUTURE WORK

By employing an ontology-based approach to specify the Bayesian networks domain, this research provided the support for knowledge intense agents to interoperate their knowledge with other agents. The resulting interoperability aids agents' specific decision-making since it facilitates the discovery of new knowledge and the exchange of information, allowing the agents to take into account evidences that were not part of their initial beliefs. Applications that demand cooperation among agents are also benefited, since agents may communicate desires to be achieved collectively.

The ontology can be applied to represent uncertain knowledge in several contexts beyond the multiagent systems, in example, Semantic Web and interchange format for Bayesian network tools and applications. The concepts and relationships were specified in such way that they can be extended to represent other probabilistic network dialects, such as Influence Diagrams.

Assuming that the interoperability is the main issue to be addressed by the ontology, the development process focuses on the conceptual modeling and on the implementation using OWL. Ontology aspects such as description logics reasoning are not addressed by this work. Beyond the interoperability, another contribution provided by the conceptual modeling perspective was the deeper understanding of the Bayesian networks domain. It was fundamental to the abstraction of the BDI model through that probabilistic knowledge representation.

The presented approach to represent uncertain knowledge, differently from PR-OWL and PACL, does not propose any modification in standards such OWL and FIPA. It was applied a current standard (OWL) to model the discrete Bayesian networks, allowing Bayesian agents (including FIPA-compliant ones) to interoperate their knowledge by adopting the ontology for defining the semantics of their messages' content.

By integrating BDI model and Bayesian networks it was obtained a BDI architecture capable of reasoning under uncertainty. It was proposed an approach to abstract the Intentional content of the mental states through Bayesian networks, enabling the association of probabilistic information within those mental states. Those probabilities are updated by the Bayesian inference process, which is triggered by the evidences perceived by the agent.

Under perceived evidences, the agents reason about their desires and take decisions about which ones to commit. This goal selection takes into account the qualitative and the quantitative aspects of the Bayesian networks. Synergistic effects can be observed in the estimative of the chances of mental states' occurrence (degree of belief to decide which desires are feasible), in the verification of compatibility between competing desires, and in the assembly of global plans involving parent variables.

The BDI agent architecture presented in the Chapter 4 do not discuss the format or structure of the action plans. The plans and the perceptions (agent's interface with its world) can be designed and implemented in accordance with the specific applications. The mental states were completely represented through Bayesian networks. Systems such as AgentSpeak(L) interpreters represent the desires and intention through plans that achieve those mental states. The BayesJason tool (an extension of the Jason tool, an AgentSpeak(L) interpreter). This work represents desires and intentions as states of world to be achieved and the plans have to be represented separately. The consideration of the causal relations (qualitative aspect of the networks) in the deliberation process enabled the agent to reason about causes and consequences. By this it means that the agents can bring about states of world that increase the chances of success.

Bringing together ontologies, Bayesian networks and the BDI agent model, gave rise to a probabilistic BDI agent architecture. The architecture is capable of interoperating its probabilistic knowledge thanks to the ontology, capable of dealing with uncertainty thanks to the Bayesian networks, and finally, capable of representing goals and taking decisions under uncertain circumstances thanks to the BDI model.

To verify the applicability of the proposal of integration, it was developed a case study in the PortEdu environment. By applying the proposed BDI agent model to the Social Agent, it was constructed the basis where cooperative tasks can be performed. Since PortEdu is a multiagent environment, inhabited by knowledge intense agents, the possibility of exchange evidences enables those agents to update their beliefs and to achieve a more reasonable behavior under uncertainty. The BDI version of the Social Agent, differently from the previous implementation, has an explicit representation of the states of affairs to be achieved. By representing beliefs about the states to be achieved, the agent can recognize when a plan has been successful. The BDI deliberation mechanism, instead of decision and utility nodes, enabled the Social Agent to check for incompatible desires, and to select desires based on feasibility and action availability.

The following research topics can be pointed as future work:

- To extend the ontology to describe Influence Diagrams. In order to represent this kind of probabilistic network, decision and utility nodes must be incorporated in the ontology. Other network models, such as Decision Trees can also be represented in the ontology by including new concepts and relationships.

- The employment of ontology alignment techniques can contribute with the improvement of the agent's perception. The belief updating process depends of a trustful perception, since the evidences have to be associated with the nodes in order to express the world's current state.

- The belief updating corresponds to the Bayesian inference, but this process is limited to update the computed marginal probabilities given the perceived evidences. To bypass that constraint it is necessary to use Bayesian network learning techniques, such as network topology learning and adjustment of probabilities in the conditional probability tables.

- Neuroscience researchers claim that the emotions play an important role in the decision making process, improving the time of response in situations that demand an acceptable result in a fashion time. From this affirmation, it is deduced that the representation of affective mental states can contribute considerably to the improvement of the deliberative process in time-constrained environments, where it is impossible to evaluate all available information. Integrating affective mental states and Bayesian Networks, it is expected an agent capable of adapting to environment uncertainties.

- The presented architecture discusses the communication (interoperability) of beliefs about evidences. But it can be used to communicate also desires and intentions. Cooperative agent systems can be developed with that framework by communicating desires and intention to be achieved collectively.

- To explore the incompatibilities among desires beyond those detected with the Bayesian networks aid. In example, spatial and temporal aspects. To take into consideration commitment strategies and intention reconsideration in the deliberation process. To represent conditional probabilities of success for plans. Beliefs about plans will help the agent to select the best ones.

- To extend the approach to the remaining agents hosted in the PortEdu. To share the ontology among the agents in the portal.

- To aggregate meaning to the chance variables and states in the Social Agent case study. It can be done by developing an ontology to specify the concepts used in the Social Agent's beliefs. Once developed the ontology, to explore this domain specific knowledge to improve the agent's cognitive processes.

7 INTEGRANDO MODELO BDI E REDES BAYESIANAS

A presente dissertação de mestrado está contextualizada na área de Inteligência Artificial, mais especificamente, na área de Agentes Autônomos e Sistemas Multiagente. Contudo, este trabalho também envolve pesquisas na área de ontologias e redes Bayesianas. O paradigma orientado a agentes provê os agentes autônomos, capazes de perceber os seus ambientes, reagir de acordo com diferentes circunstâncias e estabelecer interações sociais com outros agentes de software ou humanos. As redes Bayesianas fornecem uma maneira de representar graficamente as distribuições de probabilidades condicionais e permitem a realização de raciocínios probabilísticos baseados em evidências. As ontologias são especificações explícitas e formais de conceituações, que são usadas em uma variedade de áreas de pesquisa, incluindo os Sistemas Multiagente.

Individualmente, as linhas de pesquisa da Inteligência Artificial têm proposto abordagens para a resolução de inúmeros problemas complexos do mundo real. Contudo, existem aplicações cujos requisitos não podem ser atendidos por uma única tecnologia. Circunstâncias como estas exigem a integração de tecnologias desenvolvidas por distintas áreas da Ciência da Computação. Esta dissertação trata a integração do modelo de agentes BDI (Belief-Desire-Intention) e das redes Bayesianas. Além disso, é adotada uma abordagem baseada em ontologias para representar o conhecimento incerto dos agentes.

As ontologias têm sido usadas pela Ciência da Computação para lidar com diversas tarefas, dentre as quais citamos a modelagem conceitual e promoção de interoperabilidade. Atualmente, as pesquisas em ontologia estão fortemente relacionadas à Web Semântica (BERNERS-LEE; HENDLER; LASSILA, 2001). O propósito da Web Semântica é agregar significado às páginas Web, de modo que não somente os humanos, mas também os programas de computador possam interpretá-las. Considerando a Web Semântica um ambiente aberto e heterogêneo, habitado por agentes autônomos executando atividades em prol dos seus usuários, problemas relacionados à interoperabilidade (por exemplo, como estes agentes autônomos de domínios distintos e com objetivos distintos compartilham seu conhecimento, cooperam e maximizam a sua utilidade no sistema) surgem.

Historicamente, formalismos tradicionais para representação de conhecimento não consideram a incerteza. Exemplos dessas linguagens incluem o padrão W3C (*World Wide Web Consortium*) para a Web Semântica, o OWL (*Web Ontology Language*), e os seus predecessores XML (*eXtensible Markup Language*) e RDF (*Resource Description Framework*). Esforços para representar informações probabilísticas através de ontologias (por exemplo, o PR-OWL - *Probabilistic OWL* (COSTA; LASKEY, 2006)) têm sido feitos no contexto da Web Semântica. Contudo, esta lacuna é também percebida em outras áreas, dentre as quais podemos citar os sistemas de agentes BDI.

Maior parte das propostas de arquiteturas de agente BDI não é direcionada para lidar com representações de conhecimento incerto inerente a muitos ambientes (GEORGEFF;

INGRAND, 1989; JENNINGS et al., 1992; MÜLLER, 1996; RAO, 1996), apesar do fato que aplicações do mundo real têm que constantemente lidar com informação incerta e imprecisa. Agente com recursos limitados que habitam ambientes complexos e dinâmicos nem sempre têm acesso a informações precisas e completas. Conseqüentemente, tomar ações racionais em um tempo aceitável se torna um desafio.

7.1 Motivação

Esta dissertação foi desenvolvida no contexto do GIA (Grupo de Inteligência Artificial) da UFRGS (Universidade Federal do Rio Grande do Sul) sob a supervisão da professora Rosa Vicari. O GIA tem pesquisado e construído ambientes educacionais, usando técnicas de Inteligência Artificial para propor soluções não atendidas por outras áreas da Ciência da Computação.

O primeiro passo dado pelo presente trabalho foi motivado pela abordagem baseada em ontologias proposta por (SANTOS, 2006) para promover a interoperabilidade entre os agentes do portal PortEdu. Um dos trabalhos futuros apontados por Santos consistia em remodelar e estender a ontologia para representar a estrutura completa de uma rede Bayesiana. Estas tarefas facilitaram a abstração de estados mentais, uma vez que a ontologia permitiu um entendimento não ambíguo do referido modelo probabilístico. Através do estudo de caso, o qual consistiu em aplicar a ontologia no Agente Social (SANTOS; FAGUNDES; VICARI, 2007), foi detectada uma lacuna entre a representação de conhecimento e o comportamento orientado a objetivos. Tal lacuna motivou a integração da arquitetura BDI com as redes Bayesianas.

Durante o desenvolvimento da arquitetura BDI probabilística foi detectado que maioria dos agentes que representam seu conhecimento através de redes Bayesianas possui os processos de raciocínio dependentes das instâncias das redes. Isso impossibilita o reuso do componente de raciocínio. Redes Bayesianas compartilham uma estrutura comum (relações causais, tabelas de probabilidade condicional, variáveis de chance, estados mutuamente exclusivos), que pode ser explorada para especificar um processo deliberativo.

Finalmente, os resultados obtidos por este trabalho contribuem diretamente com as pesquisas do GIA. A integração de redes Bayesianas e arquitetura de agentes BDI será aplicada no Agente Social, um componente do PortEdu (NAKAYAMA; VICARI; COELHO, 2005). Além disso, a abordagem poderá ser estendida para os demais agentes do portal PortEdu.

7.2 Objetivos

O objetivo geral desta pesquisa é permitir que agentes BDI operem de uma maneira interoperável em ambientes onde informações incertas estão presentes. De modo a atingir o objetivo geral foram estabelecidos os seguintes objetivos específicos:

- Desenvolvimento de uma ontologia que especifica os conceitos de redes Bayesianas para atender a interoperabilidade semântica entre agentes BDI.

- Representar os estados mentais BDI através de redes Bayesianas, mais especificamente, através de elementos da ontologia de redes Bayesianas. Tal abstração é o primeiro passo em direção a um modelo BDI probabilístico.

- Especificação dos processos cognitivos da arquitetura de agentes levando em consideração que as crenças correspondem a redes Bayesianas, e desejos e intenções correspondem a estados particulares das variáveis de chance.

- Desenvolver do estudo de caso do Agente Social para demonstrar a integração das tecnologias propostas por este trabalho.

7.3 Contribuição

A contribuição mais expressiva desta pesquisa é o modelo probabilístico BDI, capaz de executar raciocínio deliberativo sob condições incertas. Diferentemente da ferramenta BayesJason (CALCIN, 2006), o qual abstrai crenças através de redes Bayesianas e estados mentais pró-ativos através de planos, este trabalho abstrai crenças, desejos e intenções através de elementos das redes Bayesianas. O processo deliberativo descrito neste trabalho usa a estrutura da representação de conhecimento e a sua semântica ao invés de instâncias particulares das redes Bayesianas. Esta abordagem permite o reuso deste componente, bem como o estudo das estruturas comuns às diferentes instâncias das redes.

Podemos afirmar que a ontologia de redes Bayesianas uma contribuição independente por si somente, uma vez que pode ser emprega em uma variedade de aplicações além da arquitetura de agente aqui apresentada. O entendimento proporcionado pela ontologia, mais especificamente pela tarefa de modelagem conceitual, facilita a integração das redes com outras tecnologias, incluindo o modelo BDI.

7.4 Representando Redes Bayesianas através de uma ontologia

Uma das principais contribuições desta dissertação é a ontologia que especifica a estrutura das redes Bayesianas. A ontologia remodela e estende os conceitos definidos em (SANTOS; BOFF; VICARI, 2006), permitindo a sua mais ampla utilização. Para guiar o desenvolvimento foi adotada a metodologia proposta por Uschold e Gruninger (USCHOLD; GRUNINGER, 1996), detalhada na Seção 2.2.3.

O principal propósito da ontologia é permitir a interoperabilidade e reuso da estrutura da representação de conhecimento de redes Bayesianas. Nesta dissertação, o escopo de aplicação foi restrito à arquitetura de agentes cognitivos, onde a ontologia atende problemas de interoperabilidade de conhecimento incerto. Os usuários alvos são desenvolvedores e pesquisadores que precisam compartilhas conceitos comuns sobre redes Bayesianas. Usuários que desejam construir modelos de outras redes probabilísticas também são considerados, uma vez que eles podem estender esta conceitualização para cobrir conceitos relacionados aos diferentes dialetos de redes probabilísticas.

A ontologia foi especificada em dois níveis. O primeiro foi em mapas conceituais, onde as restrições semânticas foram expressas parte nos mapas e parte em linguagem natural. Esta fase foi chamada captura. A segunda fase foi chamada codificação e consistiu na transição da representação capturada na fase anterior para linguagem OWL.

7.4.1 Fase de Captura

Esta fase tem como objetivo identificar os conceitos e relacionamentos do domínio das redes Bayesianas discretas, e representá-los independentemente de linguagem de codificação. O processo de captura é suportado pelos fundamentos de redes Bayesianas introduzidos na Seção 2.1.

Redes probabilísticas são modelos gráficos de interações causais entre conjuntos de variáveis, onde as variáveis são representadas por nodos de um grafo e as interações são arcos dirigidos entre nodos (COWELL et al., 1999). O grafo é a estrutura mais básica compartilhada entre redes probabilísticas. Esta estrutura, representada na Figura 7.1, é centrada no conceito de grafo (classe *Graph*). A classe *Graph* tem duas propriedades chamadas *hasNode* e *hasArc*, que respectivamente se referem a múltiplos indivíduos das classes *Node* e *Arc*. Estas duas propriedades representam componentes elementares de um grafo. É considerado que um grafo tenha ao menos um nodo. Tal restrição de cardinalidade é imposta à propriedade *hasNode* da classe *Graph*.

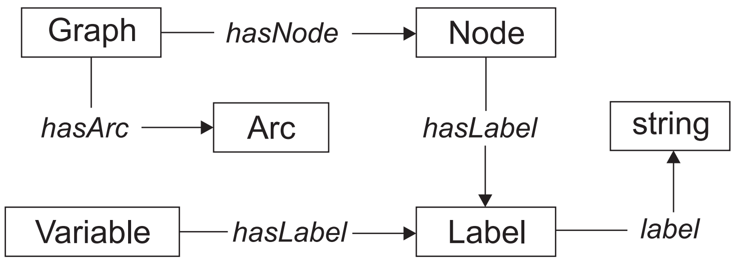


Figure 7.1: Representação de um grafo.

Os demais conceitos e relacionamentos da ontologia (redes Bayesianas discretas, situações) seguem a mesma metodologia de construção, usando relacionamentos isA para a herança de propriedades e restrições semânticas. Para maiores detalhes, ver a Seção 3.2.

7.4.2 Fase de Codificação

Conforme previamente mencionado, a presente ontologia foi codificada usando a linguagem OWL. O desenvolvimento da ontologia usou maior parte dos recursos da OWL. Os conceitos foram moldados através de classes, o relacionamento isA foi codificado como especialização, e os demais relacionamentos foram codificados como propriedades OWL. As restrições semânticas do OWL tornaram possível a contextualização dos conceitos e ajudou evitar ambigüidades dos conceitos.

Um trecho do código fonte da ontologia é ilustrado na Figura 7.2. O código começa especificando a classe *ChanceVariable*. A próxima marcação contém o comentário acerca da classe *ChanceVariable*. Seguindo, é especificada a propriedade *hasState* juntamente com a restrição de cardinalidade mínima. Por fim, é especificado que a classe *ChanceVariable* é uma subclasse de *Variable*. O código completo da ontologia é mostrado no Apêndice A.

7.5 Integrando Modelo BDI e Redes Bayesianas

Esta Seção apresenta a integração do modelo BDI com as redes Bayesianas. O efeitos sinérgicos resultantes da integração destas tecnologias da Inteligência Artificial incluem a capacidade de lidar com informação incerta em uma arquitetura BDI baseada em estados mentais.

Cada nodo em uma rede Bayesiana corresponde a exatamente uma variável de chance que tem um conjunto finito de estados mutuamente exclusivos. Afirmamos que agentes BDI, cujas crenças são representadas através de redes Bayesianas, acreditam que cada estado de uma variável de chance tem uma probabilidade de ocorrência dado um conjunto de condições impostas pelas variáveis incidentes. Supondo que os desejos correspondem

```
{...}
<owl:Class rdf:ID="ChanceVariable">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Variables representing random events. A Chance Variable
   is composed by a label and a set of states representing the
   random events.
  </rdfs:comment>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="hasState"/>
    </owl:onProperty>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
     1
    </owl:minCardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Variable"/>
  </rdfs:subClassOf>
</owl:Class>
{...}
```

Figure 7.2: Trecho do código fonte da ontologia de redes Bayesianas.

a estados de mundo que um agente deseja conferir, nosso modelo representa este estado mental através de estados das variáveis de chance que o agente deseja observar. Intenções são também representadas desse modo, uma vez que as mesmas são desejos com os quais o agente estabeleceu um comprometimento.

Supondo que as crenças do agente são representadas por redes Bayesianas, o processo de atualização de crenças corresponde à inferência probabilística. Evidências percebidas pelos agentes desempenham um importante papel no processo de atualização de crenças, uma vez que informações atuais expressam o atual estado do mundo. Então, os agentes podem reconhecer circunstâncias onde desejos são considerados atingíveis e intenções consideradas com sucesso. As informações atualizadas proporcionam suporte para o processo deliberativo, responsável por decidir quais intenções o agente deverá escolher. De modo a melhorar este processo, o agente leva em consideração os aspectos quantitativos e qualitativos das redes Bayesianas para detectar incompatibilidades e decidir entre desejos competitivos.

Para obter detalhes sobre a abstração dos estados mentais através de elementos das redes Bayesianas, consulte a Seção 4.1. Para obter detalhes sobre o processo de atualização de crenças e processo deliberativo se dirija a Seção 4.2. Aspectos de interoperabilidade dos agentes são discutidos na Seção 4.3.

7.6 Considerações Finais

Este Capítulo apresentou em língua portuguesa uma discussão geral sobre a pesquisa apresentada nesta dissertação de Mestrado. Primeiramente, o assunto foi introduzido, e logo após apresentadas motivação, objetivos e contribuições. Finalmente, foram fornecidos maiores detalhes sobre as contribuições mais significativas deste trabalho.

Através do emprego de uma abordagem baseada em ontologias para especificar as redes Bayesianas, essa pesquisa proveu o suporte para interoperabilidade entre os agentes Bayesianos. A interoperabilidade resultante auxilia o agente nos processos de tomada de decisão, uma vez que facilita a descoberta de novos conhecimentos e troca de informação, permitindo que os agentes levem em conta conhecimentos que não faziam parte das suas crenças iniciais. Aplicações que requerem cooperação também são beneficiadas, uma vez que agentes podem comunicar desejos a serem atingidos cooperativamente.

A integração do modelo BDI e das redes Bayesianas resultou em uma arquitetura BDI capaz de representar informação incerta e deliberar sob estas condições. Foi feita uma proposta para abstrair o conteúdo Intencional dos estados mentais através das redes, permitindo a associação de probabilidades condicionais aos estados mentais. Estas probabilidades são atualizadas pela inferência Bayesianas, a qual é disparada pelas percepções dos agentes. Através das evidências percebidas e conhecimento atualizado, os agentes deliberar sobre quais desejos devem ser eleitos como intenções. Esta seleção de objetivos leva em consideração os aspectos qualitativos e quantitativos das redes Bayesianas. Efeitos sinérgicos podem ser notados na estimativa de chances de ocorrência dos estados mentais (graus de crença), na verificação de compatibilidade entre desejos competitivos, e na montagem de planos globais envolvendo variáveis incidentes.

REFERENCES

ALMEIDA, V. N. **Geração de Parâmetros de Busca Baseado em Perfis de Usuário**. 2004. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**, [S.l.], v.284, n.5, 2001.

BOFF, E.; SANTOS, E. E.; VICARI, R. M. Social Agents to Improve Collaboration on an Educational Portal. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES, 2006. **Proceedings...** Los Alamitos: IEEE Computer Society, 2006. p.896–900.

BORDINI, R. H et al. (Ed.). **Multi-Agent Programming**: languages, platforms and applications. [S.l.]: Springer, 2005. (Multiagent Systems, Artificial Societies, and Simulated Organizations, v.15).

BRATMAN, M. E.; ISRAEL, D. J.; POLLACK, M. E. Plans and Resource-Bounded Practical Reasoning. **Computational Intelligence**, [S.l.], v.4, p.349–355, 1988.

BRENNER, W.; ZARNEKOW, R.; WITTIG, H. **Intelligent Software Agents**: Foundations and applications. Berlin: Springer-Verlag, 1998.

BRETON, E.; BÉZIVIN, J. Towards an understanding of model executability. In: INTERNATIONAL CONFERENCE ON FORMAL ONTOLOGY IN INFORMATION SYSTEMS, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.70–80.

BRICKLEY, D.; GUHA, R. V. **RDF Vocabulary Description Language 1.0**: rdf schema. [S.l.]: W3C, 2004. W3C Recommendation. Available at: <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. Visited on: Feb. 2007.

BROOKS, R. A Robust Layered Control System For A Mobile Robot. **IEEE Journal of Robotics and Automation RA-2**, [S.l.], p.14–23, 1986.

CALCIN, O. G. P. **Bayes Jason**. 2006. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

CARROLL, J. J.; KLYNE, G. **Resource Description Framework (RDF)**: concepts and abstract syntax. [S.l.]: W3C, 2004. W3C Recommendation. Available at: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>. Visited on: Feb. 2007.

CONNOLLY, D.; HARMELEN, F. van; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEIN, L. A. **Annotated DAML+OIL Ontology Markup**. Available at: <http://www.w3.org/TR/daml+oil-walkthru>. Visited on: Feb. 2007.

CORRêA, M. **Uma arquitetura de Diálogos entre Agentes Cognitivos Distribuídos**. 1994. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio de Janeiro, Rio de Janeiro.

COSTA, P. C. G.; LASKEY, K. B. PR-OWL: a framework for probabilistic ontologies. In: INTERNATIONAL CONFERENCE ON FORMAL ONTOLOGY IN INFORMATION SYSTEMS, 2006, Baltimore, USA. **Proceedings...** [S.l.: s.n.], 2006.

COWELL, R. G. et al. **Probabilistic Networks and Expert Systems**. [S.l.]: Springer Verlag, 1999.

DEVEDZIC, V. Understanding ontological engineering. **Commun. ACM**, [S.l.], v.45, n.4, p.136–144, 2002.

D'INVERNO, M. et al. A Formal Specification of dMARS. In: INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, ATAL, 4., 1997, Providence, Rhode Island. **Inteligent Agent IV agent Theories, Architectures, and Languages**. Berlin: Springer, 1998. p.155–176. (Lecture Notes in Computer Science, v.1365).

FENSEL, D. et al. OIL in a nutshell. In: EKAW, 12., 2000, Juan-les-Pins, France. **Knowledge Engineering and Knowledge Management. Methods, Models, and Tools.** Berlin: Springer, 2000. p.137–154. (Lecture Notes in Computer Science, v.1937).

FININ, T. et al. KQML as an Agent Communication Language. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 3., 1994, Gaithersburg, MD, USA. **Proceedings...** New York: ACM Press, 1994. p.456–463.

FIPA: Foundation For Intelligent Physical Agents. **FIPA-ACL Message Structure Specification**. Available at: <http://www.fipa.org/specs/fipa00061/SC00061G.html>. Visited on: Feb. 2007.

FIPA: Foundation For Intelligent Physical Agents. **Specifications**. Available at: <http://www.fipa.org>. Visited on: Feb. 2007.

FRANKLIN, S.; GRAESSER, A. Is it an Agent, or just a Program?: a taxonomy for autonomous agents. In: INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 3., 1996, Berlin, Germany. **Proceedings...** [S.l.]: Springer Verlag, 1996. p.21–35.

GASEVIC, D.; DEVEDZIC, V. Petri net ontology. **Knowledge Based Systems**, [S.l.], v.19, n.4, p.220–234, 2006.

GEORGEFF, M.; INGRAND, F. Decision-Making in an Embedded Reasoning System. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 11., 1989, Detroit, USA. **Proceedings...** [S.l.: s.n.], 1989. p.972–978.

GEORGEFF, M. et al. The Belief-Desire-Intention Model of Agency. In: INTERNATIONAL WORKSHOP ON INTELLIGENT AGENTS V : AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 5., 1999. **Proceedings...** Heidelberg: Springer-Verlag, 1999. p.1–10.

GLUZ, J. C.; FLORES, C. D.; SEIXAS, L.; VICARI, R. M. Formal Analysis of a Probabilistic Knowledge Communication Framework. In: IBERAMIA, 10., 2006. **Advances in Artificial Intelligence**. Berlin: Springer, 2006. p.138–148.

GUARINO, N. Understanding, Building and Using Ontologies. **International Journal of Human-Computer Studies**, [S.l.], v.46, n.2/3, p.293–310, 1997.

HORROCKS, I.; PATEL-SCHNEIDER, P.; HARMELEN, F. van. From SHIQ and RDF to OWL: the making of a web ontology language. **Journal of Web Semantics**, [S.l.], v.1, n.1, p.7–26, 2003.

HUBER, M. J. JAM: a bdi-theoretic mobile agent architecture. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 3., 1999, Seatle, WA, USA. **Proceedings...** [S.l.: s.n.], 1999. p.236–243.

JENNINGS, N. R. et al. GRATE: a general framework for cooperative problem solving. **IEE-BCS Journal of Intelligent Systems Engineering**, [S.l.], v.1, n.2, p.102–114, 1992.

JENSEN, F. et al. Hugin - The Tool for Bayesian Networks and Influence Diagrams. In: EUROPEAN WORKSHOP ON PROBABILISTIC GRAPHICAL MODELS, 1., 2002. **Proceedings...** [S.l.: s.n.], 2002.

JENSEN, F. V. **Bayesian Networks and Decision Graphs**. [S.l.]: Springer, 2001.

LUNA, J. E. O. **Algoritmos EM para Aprendizagem de Redes Bayesianas a partir de Dados Incompletos**. 2004. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Mato Grosso do Sul, Mato Grosso do Sul.

MAES, P. Artificial life meets entertainment: lifelike autonomous agents. **Communications of the ACM**, [S.l.], v.38, n.11, p.108–114, 1995.

MANOLA, F.; MILLER, E. **RDF Primer**. [S.l.]: W3C, 2004. W3C Recommendation. Available at: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>. Visited on: Feb. 2007.

MCCARTHY, J. **Ascribing mental qualities to machines**. [S.l.]: Stanford University, 1979. (STAN-CS-79-725).

MINSKY, M. **A Framework for Representing Knowledge**. [S.l.]: MIT-AI Laboratory, 1974. (Memo 306).

MÜLLER, J. P. **The Design of Intelligent Agents - A Layered Approach**. [S.l.]: Springer, 1996. (Lecture Notes in Computer Science, v.1177).

NAKAYAMA, L.; VICARI, R. M.; COELHO, H. An Information Retrieving Service for Distance Learning. **Transactions on Internet Research**, [S.l.], v.1, n.1, p.49–56, 2005.

PAN, R. et al. A Bayesian Network Approach to Ontology Mapping. In: INTERNATIONAL SEMANTIC WEB CONFERENCE, 4., 2005. **Proceedings...** [S.l.]: Springer, 2005. p.563–577. (Lecture Notes in Computer Science, v.3729).

PEARL, J. **Probabilistic reasoning in Intelligent Systems**: networks of plausible inference. [S.l.]: Morgan Kaufmann, 1988.

RAO, A. S. AgentSpeak(L): bdi agents speak out in a logical computable language. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, 7., 1996, Eindhoven, The Netherlands. **Proceedings...** [S.l.: s.n.], 1996.

SANTOS, E. R. **Uma Abordagem Baseada em Ontologias para a Interoperabilidade entre Agentes Heterogêneos**. 2006. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

SANTOS, E. R.; BOFF, E.; VICARI, R. M. Semantic Web Technologies Applied to Interoperability on an Educational Portal. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, 8., 2006. **Proceedings...** [S.l.]: Springer, 2006. p.308–317. (Lecture Notes in Computer Science, v.4053).

SANTOS, E. R.; FAGUNDES, M. S.; VICARI, R. M. An Ontology-Based Approch to Interoperability for Bayesian Agents. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 2007, Honolulu, Hawaii. **Proceedings...** New York: ACM, 2007.

SCHREIBER, A. T.; WIELINGA, B. J.; JANSWEIJER, W. H. J. The KACTUS view on the ‘O’ word. In: IJCAI WORKSHOP ON BASIC ONTOLOGICAL ISSUES IN KNOWLEDGE SHARING, 1995. **Proceedings...** [S.l.: s.n.], 1995.

SEARLE, J. R. **Intentionality**: an essay in the philosophy of mind. [S.l.]: Cambridge University Press, 1983.

SMITH, M. K.; WELTY, C.; MCGUINNESS, D. L. **OWL Web Ontology Language Guide**. [S.l.]: W3C, 2004. W3C Recommendation. Available at: <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>. Visited on: Feb. 2007.

USCHOLD, M.; GRUNINGER, M. Ontologies: principles, methods and applications. **Knowledge Engineering Review**, [S.l.], v.11, n.2, p.93–155, 1996.

VICARI, R. M.; FLORES, C. D.; SILVESTRE, A. M.; SEIXAS, L. J.; LADEIRA, M.; COELHO, H. A multi-agent intelligent environment for medical knowledge. **Artificial Intelligence in Medicine**, [S.l.], v.27, n.3, p.335–366, 2003.

WOOLDRIDGE, M. **Reasoning about Rational Agents**. Cambridge, Massachusetts: The MIT Press, 2000. (Intelligent Robots and Autonomous Agents).

WOOLDRIDGE, M.; JENNINGS, N. Intelligent Agents: theory and practice. **The Knowledge Engineering Review**, [S.l.], v.10, n.2, p.115–152, 1995.

APPENDIX A   OWL SOURCE CODE OF THE BAYESIAN NETWORK ONTOLOGY

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/Probabilistic_Networks.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.owl-ontologies.com/Probabilistic_Networks.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="ConditionalProbability">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Class that associates a probability to multiple conditions. This class is
     used to model the conditional probability of an event occurrence in the
     CPT (Conditional Probability Table).
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="probability"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasCondition"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DirectedGraph">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Probabilistic Networks are graphical models of causal interactions among
     a set of variables, where the variables are represented as nodes of a graph
     and the interactions as directed arcs between the nodes.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="DirectedArc"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasArc"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Graph"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="ChanceVariable">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Variables representing random events. A Chance Variable is composed by a
     label and a set of states representing the random events.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasState"/>
        </owl:onProperty>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Variable"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="UtilityNode">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Node"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#DirectedArc">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="hasChild"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Arc"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A Directed Arc is a link from a child node to a parent node. A Directed Arc
     is represented by an arrow from parent node to child node.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasParent"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Node">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A node represents variables or utility functions. In models containing
     neither decision variables nor utility functions, the concepts of Node
     and Variable can be used interchangeably. For models that contain decision
     variables and utility functions it is convenient distinguish between
     nodes and variables, as a node does not necessarily represents a variable.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="hasLabel"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:ID="BayesianSituation">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="BayesianNetwork"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasGraph"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A bayesian situation is a situation that can reference only bayesian
     networks.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Situation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Evidence">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasNode"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     An Evidence is an information received from external sources about the
     state or value of a variable of a probabilistic network. The information
     contained in an evidence modifies the chances associated with a variable.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="ChanceNode"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#hasLabel"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="#probability"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="#ChanceNode">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
```

```
   This category of Node represents a chance variable. It is represented
   graphically by a labeled circle.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Node"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="hasChanceVariable"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:ID="ConditionalChanceVariable"/>
            <owl:Class rdf:ID="PriorChanceVariable"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#hasChanceVariable"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="BayesianArc">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   A Bayesian Arc is a specialization of a Directed Arc that only connects
   chance nodes.
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#ChanceNode"/>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#hasChild"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#DirectedArc"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasParent"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#ChanceNode"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="JunctionTree">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="UndirectedGraph"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasArc"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="SeparatorArc"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="CliqueNode"/>
      </owl:allValuesFrom>
      <owl:onProperty>
```

```xml
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#SeparatorArc">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#CliqueNode"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasSeparator"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="UndirectedArc"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="BayesianSituationTransition">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A Bayesian Situation Transition is a specific kind of transition that
     occurs only in Bayesian Networks.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="SituationTransition"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#BayesianNetwork">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Bayesian Networks are Probabilistic Networks that contain only random
     variables (chance nodes) and directed arcs that represent direct
     dependencies among the variables.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#ChanceNode"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#DirectedGraph"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasArc"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#BayesianArc"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DecisionVariable">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Variable"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Variable">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#hasLabel"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   A Variable represents an exhaustive set of mutually exclusive events,
   referred to as the domain of the variable.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="HardEvicence">
  <rdfs:subClassOf rdf:resource="#Evidence"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#probability"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   An Evidence that assigns one to the probability property. The indicated state
   has been observed.
  </rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="ConditionalState">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="State"/>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   A Conditional State has multiple conditional probabilities. Each conditional
   probability specifies the conditions and the state chance of occurrence
   under these conditions. This class corresponds to a line in a CPT.
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasConditionalProbability"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Arc">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   An Arc is a link between two nodes.
  </rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#Situation">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasGraph"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   A situation corresponds to a particular configuration of a probabilistic
   network given a set of evidences.
  </rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#CliqueNode">
  <rdfs:subClassOf>
```

```
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasClique"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Node"/>
  </owl:Class>
  <owl:Class rdf:about="#PriorChanceVariable">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A Prior Chance Variable corresponds to a variable of a prior node. It
     has associated multiple individuals of State Probabilities class. Each
     state probability denotes a state and its chance.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="StateProbability"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasState"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#ChanceVariable"/>
  </owl:Class>
  <owl:Class rdf:ID="Condition">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#ChanceNode"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A Condition is a specification of a particular state in a chance node variable.
     The state was explicited since it is necessary in order to differentiate from
     others states from the same node.
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasState"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#UndirectedArc">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A Undirected Arc is a undirected link between two nodes.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasNode"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         2
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Arc"/>
  </owl:Class>
  <owl:Class rdf:about="#State">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#hasLabel"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     States are also often called events, levels, values, choices or options.
     They are associated with a variable and they are mutually exclusive.
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="#ConditionalChanceVariable">
    <rdfs:subClassOf rdf:resource="#ChanceVariable"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#ConditionalState"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasState"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A Conditional Chance Variable corresponds to a CPT (Conditional Probability
     Table). The states are references to individual of Conditional State class
     which denotes the state and its conditional probabilities. Each individual
     in the state property is an abstraction of a line in a CPT, supposing each
     line represents a state of the Conditional Chance Variable.
    </rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#StateProbability">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     A State Probability is the chance of occurrence of a variable state. It is
     composed by the state label and the state chance of occurrence.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
         1
        </owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="#probability"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#State"/>
  </owl:Class>
  <owl:Class rdf:about="#UndirectedGraph">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Graph"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasArc"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#UndirectedArc"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:ID="InfluenceDiagram">
  <rdfs:subClassOf rdf:resource="#DirectedGraph"/>
</owl:Class>
<owl:Class rdf:about="#Graph">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasNode"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="DecisionNode">
  <rdfs:subClassOf rdf:resource="#Node"/>
</owl:Class>
<owl:Class rdf:about="#SituationTransition">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasPosteriorSituation"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   A Situation Transition occurs because an evidence has been received. These
   evidences cause modifications in the chances of some variables in the
   network. These modifications are executed in the network referenced by
   prior situation and the updated network is stored in the posterior
   situation.
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="hasPriorSituation"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
       1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Label">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   A Label is used to denote a name.
  </rdfs:comment>
</owl:Class>
<owl:ObjectProperty rdf:about="#hasState">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#ChanceVariable"/>
        <owl:Class rdf:about="#Condition"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes one or more states associated with a Chance Variable.
   A state is also called level, value, choice, or option.
  </rdfs:comment>
  <rdfs:range rdf:resource="#State"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEvidence">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes a set of evidences.
  </rdfs:comment>
  <rdfs:range rdf:resource="#DirectedGraph"/>
  <rdfs:domain rdf:resource="#Situation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasSeparator">
  <rdfs:domain rdf:resource="#SeparatorArc"/>
  <rdfs:range rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasGraph">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes the probabilistic network individual that
   represents the actual situation of the network.
  </rdfs:comment>
  <rdfs:domain rdf:resource="#Situation"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="#Graph"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasArc">
  <rdfs:range rdf:resource="#Arc"/>
  <rdfs:domain rdf:resource="#Graph"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasConditionalProbability">
  <rdfs:range rdf:resource="#ConditionalProbability"/>
  <rdfs:domain rdf:resource="#ConditionalState"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes a set of individuals of class Conditional Probability
   associated with a particular state in a non prior node variable.
  </rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasCondition">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes a set of conditions.
  </rdfs:comment>
  <rdfs:range rdf:resource="#Condition"/>
  <rdfs:domain rdf:resource="#ConditionalProbability"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasClique">
  <rdfs:domain rdf:resource="#CliqueNode"/>
  <rdfs:range rdf:resource="#UndirectedGraph"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasPosteriorSituation">
  <rdfs:domain rdf:resource="#SituationTransition"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="#Situation"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes a situation after a notification of an evidence.
  </rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasMarginalDistribution">
  <rdfs:domain rdf:resource="#ChanceVariable"/>
  <rdfs:range rdf:resource="#StateProbability"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property that denotes the computed marginal distribution of a variable. It
   is composed by multiple state probabilities.
  </rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasNode">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Property of a Probabilistic Network that denotes its set of nodes.
  </rdfs:comment>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Condition"/>
        <owl:Class rdf:about="#Evidence"/>
        <owl:Class rdf:about="#UndirectedArc"/>
        <owl:Class rdf:about="#Graph"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasParent">
  <rdfs:domain rdf:resource="#DirectedArc"/>
```

```
    <rdfs:range rdf:resource="#Node"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Property of a Directed Arc that denotes the parent node.
    </rdfs:comment>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="label">
    <rdfs:domain rdf:resource="#Label"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Property that denotes a name or an identification.</rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#probability">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Property that denotes a probability associated with a variable state.
    </rdfs:comment>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#StateProbability"/>
          <owl:Class rdf:about="#ConditionalProbability"/>
          <owl:Class rdf:about="#Evidence"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:about="#hasPriorSituation">
    <rdfs:range rdf:resource="#Situation"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Property that denotes a situation before a notification of an evidence.
    </rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#SituationTransition"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#hasLabel">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Property that denotes an individual of the class Label.
    </rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Variable"/>
          <owl:Class rdf:about="#Node"/>
          <owl:Class rdf:about="#State"/>
          <owl:Class rdf:about="#Evidence"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#Label"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#hasChanceVariable">
    <rdfs:range rdf:resource="#ChanceVariable"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Property that denotes the variable associated with a node. A prior node
     has a Chance Variable and a not prior node is defined by multiple Conditional
     Chance Variable, which represents a Conditional Probability Table (CPT).
    </rdfs:comment>
    <rdfs:domain rdf:resource="#ChanceNode"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#hasChild">
    <rdfs:domain rdf:resource="#DirectedArc"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Property of a Directed Arc that denotes the child node.
    </rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Node"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

APPENDIX B PAPER ACCEPTED IN THE CONFERENCE AAMAS 2007

The paper "An Ontology-Based Approach to Interoperability for Bayesian Agents" was submitted and accepted as a short paper in the Sixth International Conference on Autonomous Agents and Multiagent Systems, to be held May 14–18 2007 in Honolulu, Hawaii.

An Ontology-Based Approach to Interoperability for Bayesian Agents

Elder Rizzon Santos
ersantos@inf.ufrgs.br

Moser Silva Fagundes
msfagundes@inf.ufrgs.br

Rosa Maria Vicari
rosa@inf.ufrgs.br

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Po. B 15.064 - 91.501-970 - Porto Alegre - RS - Brazil. Phone/Fax: +55 51 33166161

ABSTRACT

This paper presents an ontology-based approach to promote the interoperability among agents that represent their knowledge through Bayesian networks. This research relies on semantic web foundations to achieve knowledge interoperability in the context of multiagent systems. Our first step was the specification of an ontology that formalizes the structures of the Bayesian network representation. Once handled the issue of the knowledge representation, we specify how a Bayesian agent operates such representation. Thus, we define a model of internal architecture to support Bayesian agents in the knowledge sharing and maintenance tasks.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods —*semantic networks*

General Terms

Design, Experimentation, Standardization

Keywords

semantic web, ontology, interoperability, bayesian networks, agent architecture

1. INTRODUCTION

Studies on interoperability on the context of Artificial Intelligence have been done mostly for the communication among intelligent agents. Today, such researches can be applied for the development of the semantic web, which is the mainstream on Internet technology. Considering the semantic web as an open system, populated by autonomous agents carrying out activities in behalf of its owners, interoperability issues (i.e. how these agents from different domains and with different goals will share their knowledge, co-operate and maximize the utility of the whole system) arise.

This paper presents an agent architecture that allows the interoperability of knowledge among Bayesian agents. Specifically, we are concerned on how heterogeneous Bayesian agents may exchange their knowledge. We consider Bayesian agents [3] those that have their knowledge expressed through Bayesian networks. The fact that the agents use the same knowledge representation (i.e. Bayesian networks) does not guarantee that it is implemented in an interoperable way.

2. BAYESIAN NETWORK ONTOLOGY

One of the contributions of this research is the specification of an ontology to formalize the Bayesian network knowledge representation. Our ontology specification extended the concepts defined in [2], allowing a more broad utilization of the ontology. One of these utilizations is an agent architecture for interoperability, described in the section 3.

A discrete Bayesian network consists of a DAG (Directed Acyclic Graph) and a set of conditional probability distributions [1]. Each node in the network, called chance node, corresponds to exactly one discrete random variable which has a finite set of mutually exclusive states. The directed arcs specify the causal relation between the random variables. Each random variable associated with a chance node has a conditional probability distribution.

The *BayesianNetwork* class (Figure 1) is the core of our Bayesian network definition. This concept has two properties: *hasArc* and *hasNode*, which respectively refer multiple individuals of classes *ChanceNode* and *BayesianArc*.

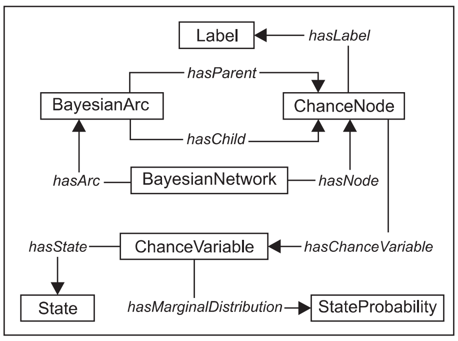


Figure 1: Bayesian Network representation.

A *BayesianArc* individual defines a link between two chance nodes. It has the *hasParent* and *hasChild* properties. The value of these properties is an individual of the *ChanceNode* class. A *ChanceNode* individual has a chance variable associated to its definition. Such variable is specified by the *hasChanceVariable* property. This property allows only individuals of the classes *PriorChanceVariable* and *ConditionalChanceVariable*. Such constraint is necessary in order to differentiate prior nodes variables from non-prior nodes.

Before defining a chance variable it is necessary to define a state and its related concepts (Figure 2). A state is represented by the *State* class, which has only the *hasLabel* property responsible for the node identification. The *State* class has two direct subclasses. The first denotes a chance associated with a state and it is called *StateProbability*. It is defined by the inherited *hasLabel* property and the *probability* property (float data type). The second specialization is named *ConditionalState* and it specifies the multiple conditional chances associated with a state. A set of *ConditionalState* individuals constitutes a Conditional Probability Table (CPT). The *ConditionalState* class has two properties: the inherited *hasLabel*, which represents the label of the state and the *hasConditionProbability* property, which references multiple individuals of the *ConditionProbability* class.

The *ConditionProbability* class represents the conditional chances associated with a state. This class is defined by the probability and the *hasCondition* properties. The former is a float data type property that represents the numerical *probability* of a variable's state under the conditions specified in the *hasCondition* property. This property references multiple individuals of the *Condition* class, and it denotes the conditions imposed in the probability of a state. The *Condition* class is constituted by a conditioning node and a state of this node, respectively referenced by the properties *hasNode* and *hasState*. The individual referenced by the *hasNode* property must be a *ChanceNode* since only chance nodes have random variables. The *hasState* property references an individual of the *State* class that indicates the specific state of the conditioning chance variable.

The *ChanceVariable* class represents a set of mutually exclusive states. The states, also called events or choices, correspond to the domain of the variable, which can be discrete or continuous. In this work we consider only discrete variables (finite sets). The *ChanceVariable* class specifies the *hasLabel*, *hasState* and *hasMarginalDistribution* properties. The first property identifies and provides an unique name for the variable. The second specifies the necessity of at least one state (represented by *State* individuals) associated with a variable (i.e. true or false in the context of a boolean variable). The last property represents the computed marginal distribution for the chance variable, and it references multiple *StateProbability* individuals. Each individual represents a state and its computed chance of occurrence.

It was necessary to differentiate prior node variables from non-prior ones, since a non-prior node has a CPT, and a prior node has only states and probabilities without conditioning variables. Thus, the classes *PriorChanceVariable* and *ConditionalChanceVariable* were created as subclasses of *ChanceVariable*. The difference between these two subclasses lies in the *hasState* property constraint. In the *PriorChanceVariable* class the *hasState* property has been restricted and it can reference only *StateProbability* individuals. As stated earlier, a state probability represents a state and its chance of occurrence. The set of *StateProbability* individuals referenced by the *hasState* property denotes all

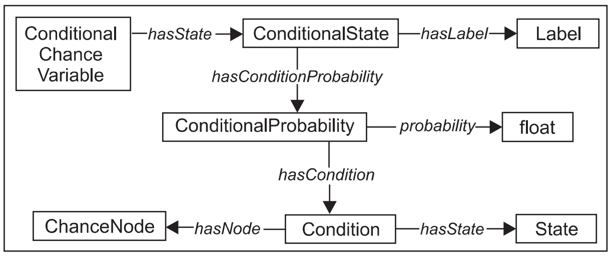


Figure 2: Conditional Probability Table representation.

possible states associated with a prior chance variable. The *hasState* property of *ConditionalChanceVariable* class has also been constrained and only *ConditionalState* individuals can be assigned to it. The *ConditionalState* individuals represent a Conditional Probability Table of a variable associated with a non-prior node.

In our definition, a situation is a particular configuration that a probabilistic network assumes given a set (possibly empty) of evidences of events occurrence. When the evidences of events are reflected in the network, a new situation arises. Such situations are useful to keep the history of modifications of a Bayesian network.

An evidence, represented by the *Evidence* class, corresponds to any information regarding the state of a variable from a probabilistic network. The *Evidence* class is composed by a node, a label and a chance, represented by properties *hasNode*, *hasLabel* and *probability*, respectively. The *hasNode* property can reference only individuals of the *ChanceNode* class, since chance nodes are the only kind of node that represent random events.

A situation, represented by the *Situation* class, has two properties. The first is the *hasBayesianNetwork* property used to reference the network individual whose configuration corresponds to the given situation. The second property is the *hasEvidence* that corresponds to the set of evidences that originates the situation.

In order to establish a link between two sequential situations we created a class named *SituationTransition*. This class is described by *hasPriorSituation* property and *hasPosteriorSituation* property, which represents a prior and a posterior situation, respectively.

3. BAYESIAN AGENT INTERNAL ARCHITECTURE

The agent architecture for interoperability presented in this section extends and generalizes the architecture proposed in [2]. The main goal of our architecture is to enhance the interoperability of Bayesian network knowledge among agents. The interoperability is achieved by an ontology-based approach to represent the uncertain knowledge of the agent. Following, we detail the agents' internal architecture, depicted in the Figure 3.

The *Agent Implementation Specific Components* are not specified by this architecture since they relate to the particular purpose of each agent design. However, we specify the way they interact with the other architecture components. Usually, the *Agent Implementation Specific Components* define the manner that the agent reasons about its goals and how it achieves them (i.e. planning and goal de-

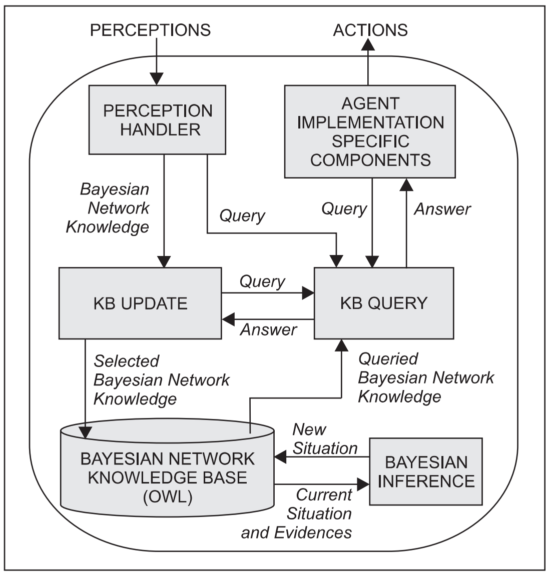

PERCEPTIONS
ACTIONS
PERCEPTION HANDLER
AGENT IMPLEMENTATION SPECIFIC COMPONENTS
Bayesian Network Knowledge
Query
Query
Answer
KB UPDATE
Query
Answer
KB QUERY
Selected Bayesian Network Knowledge
Queried Bayesian Network Knowledge
New Situation
BAYESIAN NETWORK KNOWLEDGE BASE (OWL)
BAYESIAN INFERENCE
Current Situation and Evidences

Figure 3: Bayesian Agent Internal Architecture.

liberation). The first component of the architecture is the *Perception Handler*, which receives and forwards the perceptions to the respective components capable of interpreting them. The characteristics of a perception are taken into account to decide which component will receive it. Since in the context of this work we are dealing with interoperability among Bayesian agents, we focus on two particular categories of perception: *Bayesian Network Knowledge* and *Query*. The first corresponds to individuals of the ontology presented in the section 3. The perceptions of this category are forwarded to the *Knowledge Base (KB) Update* component. The second corresponds to queries about the agent's knowledge that are forwarded to the *KB Query* component.

The second component of the architecture is the *KB Update*. Its purpose is to evaluate the incoming *Bayesian Network Knowledge*, and insert the selected ones in the knowledge base as individuals of the Bayesian network ontology. The information to be inserted is selected following the criteria defined by the designer. A simple implementation of this component performs insertions in the KB without restrictions. A more sophisticated implementation interacts with the *KB Query* component to retrieve already inserted Bayesian information to constrain the information to be inserted.

Our knowledge base is constituted by the Bayesian network ontology, detailed in the section 2, and its individuals. It stores the Bayesian networks situations, the transitions between situations and the evidences. The base can contain multiple different Bayesian networks. Any modification in a Bayesian network characterizes a new situation, and the sequence of situations represents a history of a network. The history may be useful for an agent planning, in example.

In order to perform probabilistic reasoning in the Bayesian networks stored in the knowledge base, we specify the *Bayesian Inference* component. Its inputs are the *Current Situation* of a Bayesian network and a set of *Evidences*. The *Bayesian Inference* output is the *New Situation* with its probabilities recalculated considering the *Evidences*. It is worth to point out that both situations are individuals of the *Bayesian-Network* class and that the *Evidences* are *Evidence* class individuals. The *New Situation* resulting from the inference process constitutes the most up-to-date knowledge that the agent has about its domain. The presence of this component is indispensable since updated knowledge is necessary to support the agent decisions and actions.

The *KB Query* component receives queries from *Agent Implementation Specific Components*, *Perception Handler* and *KB Update*. These queries can return events (states) and their occurrence probabilities, causal relations between variables and other information that can be inferred from the Bayesian networks knowledge base. Queries from the agent specific components usually are performed to aid the agent in its decision making process. The queries forwarded by *Perception Handler* are related to knowledge that external agents need to be informed about. Finally, the queries from the *KB Update* component are executed with the purpose of selecting which information will be inserted on the KB.

The Agent Implementation Specific Components are not specified by the architecture since they relate to the particular purpose of each agent design. however, we specify the way they interact with the architecture components. Usually, this component define the manner that the agent reasons about its goals and how it achieves them (i.e. planning and goal deliberation).

The core of the interoperability relies on the Bayesian network ontology. It provides the fundamental domain concepts among the Bayesian agents making possible their knowledge exchange. The architecture supports the knowledge representation in a broader way, not only in the interaction with other Bayesian agents. The architecture also provides the means for the knowledge maintenance.

4. CONCLUSION

We define an internal architecture that provides support for knowledge intense agents to interoperate their knowledge with other agents. In this case, the interoperation is in the scope of Bayesian knowledge regarding an adequate way to express it. Besides that, we provide resources for maintaining such Bayesian knowledge. Maintenance features allow the execution of updates, queries and an inference process to propagate evidences to the corresponding networks present in the knowledge base. The interoperability provided by this architecture aids agent specific decision making since it facilitates the discovery of new knowledge, allowing the agent to consider evidences that were not part of its original KB.

5. REFERENCES

[1] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer Verlag, 1999.

[2] E. R. Santos, E. Boff, and R. M. Vicari. Semantic web technologies applied to interoperability on an educational portal. In *Intelligent Tutoring Systems, 8th International Conference*, volume 4053, pages 308–317. Springer, 2006.

[3] R. M. Vicari, C. D. Flores, A. M. Silvestre, L. J. Seixas, M. Ladeira, and H. Coelho. A multi-agent intelligent environment for medical knowledge. *Artificial Intelligence in Medicine*, 27(3):335–366, 2003.

APPENDIX C PAPER ACCEPTED IN THE WORKSHOP PROMAS 2007

The paper "Interoperability for Bayesian Agents in the Semantic Web" was submitted and accepted in the Programming Multi-Agent Systems Workshop (ProMAS), to be held with the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007).

Interoperability for Bayesian Agents in the Semantic Web

Elder Rizzon Santos, Moser Silva Fagundes, and Rosa Maria Vicari

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Po. B 15.064 - 91.501-970 - Porto Alegre - RS - Brazil. Phone/Fax: +55 51 33166161
{ersantos,msfagundes,rosa}@inf.ufrgs.br

Abstract. This paper presents an ontology-based approach to promote the interoperability among agents that represent their knowledge through Bayesian networks. This research relies on Semantic Web foundations to achieve knowledge interoperability in the context of multiagent systems. Our first step was the specification of an ontology that formalizes the structures of the Bayesian network representation. It was developed using OWL, which is a W3C recommendation for ontology language. Once handled the issue of the knowledge representation, we specify how a Bayesian agent operates such representation. Thus, we define a model of internal architecture to support Bayesian agents in the knowledge sharing and maintenance tasks. The utilization of the architecture is exemplified through a case study developed in the context of a multiagent educational portal (PortEdu). The case study demonstrates the interoperability resulted from the architecture integration with Bayesian agents hosted in PortEdu.

Key words: Semantic Web, Ontology, Interoperability, Bayesian Networks, Agent Architecture

1 Introduction

Studies on interoperability on the context of Artificial Intelligence have been done mostly for the communication among intelligent agents [1, 2]. Today, such researches can be applied for the development of the Semantic Web [3], which is the mainstream on Internet technology. The purpose of the Semantic Web is to aggregate meaning to web pages, in a way that not only humans, but also computer software may interpret its content. Considering the Semantic Web as an open system, populated by autonomous agents carrying out activities in behalf of its owners, interoperability issues (i.e. how these agents from different domains and with different goals will share their knowledge, co-operate and maximize the utility of the whole system) arise.

This paper presents an agent architecture that allows the interoperability of knowledge among Bayesian agents. We consider Bayesian agents those that have their knowledge expressed through Bayesian networks. The fact that the agents use the same knowledge representation (i.e. Bayesian networks) does not guarantee that it is implemented in an interoperable way.

Our case study is contextualized in a multiagent system (MAS) that supports agent-based educational systems. This MAS, called PortEdu [4], is a FIPA (Foundation for Intelligent Physical Agents) [2] compliant agent platform that provides infrastructure and services for the systems in the portal. One of these services is provided by the Social Agent [5], responsible for organizing the users in groups considering cognitive and emotional aspects. Such social aspects are represented by the agent as Bayesian networks. The case study consists in applying the proposed architecture in the Social Agent.

The interoperability core relies on the specified ontology for Bayesian networks. Ontologies are used to provide means to formalize concepts and the relationships among them, allowing agents to interpret their meaning flexibly and unambiguously [6]. In open systems, such as the Semantic Web, it is necessary to have a standard way to communicate the knowledge. The W3C (World Wide Web Consortium) is developing a set of recommendations to deal with this issue. One of them is OWL (Web Ontology Language) [7, 8]. It is designed specifically for the purpose of knowledge communication in the Semantic Web. Currently, it is considered the standard for content languages to be adopted in the Semantic Web. Relying on a standard for communication solves an important interoperability issue, the agreement on a well defined and common language.

The remaining of this paper is organized as follows: section 2 presents the related researches; section 3 specifies the Bayesian network ontology; in section 4 we describe the Bayesian agent internal architecture to interoperability; section 5 presents the application of the architecture on the social agent; and in section 6 it is presented our conclusions and future work.

2 Related Research

BayesOWL [9] was developed to handle the issue of automatic ontology mapping. This approach defines additional markups that can add probabilities to concepts, individuals, properties and its relationships. It also defines a set of translation rules to convert the probabilistic annotated ontology into a Bayesian network. The focus on ontology mapping limits the BayesOWL markups since it was not necessary to represent variables with states different than true or false. The reason for this is that the probabilistic knowledge associated with each ontology concept was used only for telling if two concepts from different ontologies were the same.

Another approach that represents probabilistic knowledge through OWL is PR-OWL [10]. Its goal is to provide a framework for probabilistic ontologies. It constitutes an extension of OWL to express probabilistic knowledge. The PR-OWL language adds new definitions to OWL allowing the expression of uncertainty. The need for standardization represents a drawback for short-term solutions but also points to a very interesting medium to long term solution, as it fits well (providing the formal foundation of a first-order logic) in the W3C model of standards.

The objective of [11] is to provide the necessary structure to share petri nets on the Semantic Web context. This work reviews previous efforts done in petri net sharing and petri net formalizations. Then, it specifies a petri net ontology using OWL language. Another work concerning petri net representation is [12]. Its main goal is the understanding of the model executability. In order to achieve this goal, it discusses petri net related concepts, classifying them in static or dynamic. The final result is a three level petri net metamodel. The first level is the definition metamodel that specifies the static part of the nets. The second level defines a particular situation of a petri net. The third level is an execution metamodel that defines a sequence of situations.

Agent communication issues regarding probabilities are addressed in [13], where is presented PACL (Probabilistic Agent Communication Language). It is an extension of the FIPA-ACL designed to deal with the communication of probabilistic knowledge. PACL specifies new communication axioms that are necessary to model the probabilistic communication. Besides the axioms, the language also designs assertive and directive probabilistic speech acts, which extends FIPA-ACL. The PACL language provides a way to communicate probabilistic knowledge extending FIPA-ACL and allowing more expressiveness to this language. It does not deals with the communication of uncertainty at the message content level, concerning how different Bayesian agents might exchange knowledge regarding their networks and evidences.

In [14] it is described an approach to promote interoperability providing a conversion engine based on OWL. The work defines an ontology that covers elementary aspects necessary to construct Bayesian networks individuals, but it does not formalizes the Bayesian network knowledge representation. The conversion engine uses the ontology to automatically generate individuals from a Bayesian network implementation following a standard format. The resulting knowledge base is part of an architecture to promote interoperability for a specific agent.

3 Bayesian Network Ontology

One of the contributions of this research is the specification of an ontology to formalize the Bayesian network knowledge representation. Our ontology specification extends the concepts defined in [14], allowing a broader utilization of the ontology. One of these utilizations is an agent architecture for interoperability, described in the Section 4.

In the following sub-sections we provide a detailed description of the developed ontology to represent Bayesian knowledge. First, we present the common concepts among different probabilistic networks and a specialization of this knowledge presenting the discrete Bayesian network definitions. Then, we detail the evidence-related concepts and their relation to the evolution of the Bayesian network individuals.

The figures in this section illustrates concept maps of the classes specified in the ontology. This graphical representation provides a way to visualize the classes (depicted as rectangles) and the relationship among them (illustrated as arrows).

3.1 Probabilistic Network Concepts

Probabilistic networks are graphical models of causal interactions among a set of variables, where the variables are represented as nodes of a graph and the interactions as directed arcs between nodes [15].

A graph is the basic structure shared between probabilistic network models. It is formalized in the ontology by the *Graph* class (Figure 1a). It has two properties named *hasNode* and *hasArc*, which respectively reference multiple individuals of classes *Node* and *Arc*. These two properties represent elementary components of a graph. We are considering that a graph has at least one arc and at least two nodes. Such cardinality constraints are imposed in the *Graph* properties *hasArc* and *hasNode*.

The *ProbabilisticNetwork* class (Figure 1b) represents a probabilistic network and it is a subclass of *Graph*. The *ProbabilisticNetwork* class models common aspects among variations in this kind of knowledge representation (i.e. Bayesian networks, influence diagrams and object-oriented probabilistic networks). Those common elements are the directed arcs and the nodes, respectively referenced by the inherited properties *hasArc* and *hasNode*. The *hasArc* property is semantically restricted to reference only *DirectedArc* class individuals.

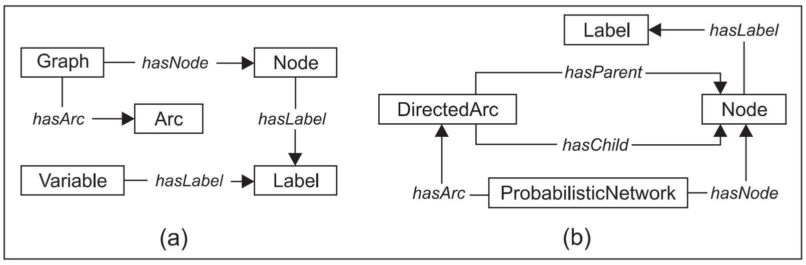


Fig. 1. (a) Graph and (b) Probabilistic Network representations.

To specify the direct link between a parent and a child node we define the *DirectedArc* class, a specialization of the *Arc* class. Such link is represented through the properties *hasChild* and *hasParent* of the arc class. The value of these properties is an individual of the *Node* class. This class is defined by a unique label, denoted by the *hasLabel* property. The label is the common aspect among chance, decision and utility nodes. Since we are dealing with Bayesian networks we specify only chance nodes.

Another general concept concerning probabilistic networks is defined by the *Variable* class. It represents a set of mutually exclusive states. The states, also called events or choices, correspond to the domain of the variable, which can be discrete or continuous. In this work we consider only discrete variables (finite sets). A probabilistic network has two categories of variable: chance variables, representing random states, and decision variables, representing choices controlled by some agent. As the *Node* class, the *Variable* has only the *hasLabel* property. The purposes of the *Variable* class are the same of the *Node* class, which are to abstract the complexity and model common aspects between its subclasses.

3.2 Discrete Bayesian Network Concepts

A discrete Bayesian network consists of a DAG (Directed Acyclic Graph) and a set of conditional probability distributions [16]. Each node in the network, called chance node, corresponds to exactly one discrete random variable which has a finite set of mutually exclusive states. The directed arcs specify the causal relation between the random variables. Each random variable associated with a chance node has a conditional probability distribution.

The *BayesianNetwork* class (Figure 2) is the core of our Bayesian network definition. It is a subclass of *ProbabilisticNetwork*. The differences among the classes are the semantic constraints imposed to the properties to specify the correct type of nodes and arcs allowed in a Bayesian network. Such kinds of nodes and arcs are represented by the *ChanceNode* and *BayesianArc* classes respectively.

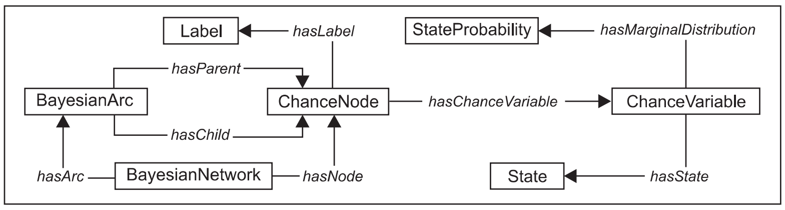


Fig. 2. Bayesian Network representation.

A *BayesianArc* individual defines a link between two chance nodes. It inherits the *hasParent* and *hasChild* properties from the *DirectedArc* class, and imposes additional constraints to formalize that only individuals of the *ChanceNode* class can be assigned to these properties. A *ChanceNode* individual has a chance variable associated to its definition. Such variable is specified by the *hasChanceVariable* property. This property allows only individuals of the classes *PriorChanceVariable* and *ConditionalChanceVariable*. Such constraint is necessary in order to differentiate prior nodes variables from non-prior nodes.

Before defining a chance variable it is necessary to define a state and its related concepts (Figure 3). A state is represented by the *State* class, which has only the *hasLabel* property responsible for the node identification. The *State* class has two direct subclasses. The first denotes a chance associated with a state and it is called *StateProbability*. It is defined by the inherited *hasLabel* property and the *probability* property (float data type). The second specialization is named *ConditionalState* and it specifies the multiple conditional chances associated with a state. A set of *ConditionalState* individuals constitutes a Conditional Probability Table (CPT). The *ConditionalState* class has two properties: the inherited *hasLabel*, which represents the label of the state and the *hasConditionProbability* property, which references multiple individuals of the *ConditionProbability* class.

The *ConditionProbability* class represents the conditional chances associated with a state. This class is defined by the *probability* and the *hasCondition* properties. The former is a float data type property that represents the numerical *probability* of a variable's state under the conditions specified in the *hasCondition* property. This property references multiple individuals of the *Condition* class, and it denotes the conditions imposed in the probability of a state. The *Condition* class is constituted by a conditioning node and a state of this node, respectively referenced by the properties *hasNode* and *hasState*. The individual referenced by the *hasNode* property must be a *ChanceNode* since only chance nodes have random variables. The *hasState* property references an individual of the *State* class that indicates the specific state of the conditioning chance variable.

The *ChanceVariable* class is a specialization of *Variable* and it represents a chance variable. Additionally to the inherited properties from the *Variable* class, it specifies the *hasState* and *hasMarginalDistribution* properties. The first specifies the necessity of at least one state (represented by *State* individuals) associated with a variable (i.e. true or false in the context of a boolean variable). The second property represents the computed marginal distribution for the chance variable, and it references multiple *StateProbability* individuals. Each individual represents a state and its computed chance of occurrence.

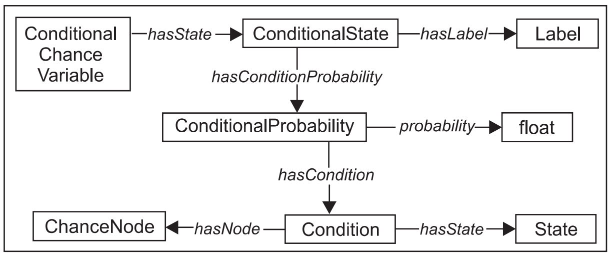


Fig. 3. Conditional Probability Table representation.

It was necessary to differentiate prior node variables from non-prior ones, since a non-prior node has a CPT, and a prior node has only states and probabilities without conditioning variables. Thus, the classes *PriorChanceVariable* and *ConditionalChanceVariable* were created as subclasses of *ChanceVariable*. The difference between these two subclasses lies in the *hasState* property constraint. In the *PriorChanceVariable* class the *hasState* property has been restricted and it can reference only *StateProbability* individuals. As stated earlier, a state probability represents a state and its chance of occurrence. The set of *StateProbability* individuals referenced by the *hasState* property denotes all possible states associated with a prior chance variable. The *hasState* property of *ConditionalChanceVariable* class has also been constrained and only *ConditionalState* individuals can be assigned to it. The *ConditionalState* individuals represent a Conditional Probability Table of a variable associated with a non-prior node.

The *hasLabel*, a common property among *Node*, *Variable* and *State* concepts, has as default value a *Label* class individual. The *Label* class is composed just by a string indicating the label name. However, the *hasLabel* property is also able to indicate individuals of other classes, which enables the developers to add semantics to those concepts.

3.3 Situation Concepts

In our definition, a situation is a particular configuration that a probabilistic network assumes given a set (possibly empty) of evidences of events occurrence. When the evidences of events are reflected in the network, a new situation arises. Such situations are useful to keep the history of modifications of a Bayesian network. The Figure 4 depicts the situation related concepts.

An evidence, represented by the *Evidence* class, corresponds to any information regarding the state of a variable from a probabilistic network. The *Evidence* class is composed by a node, a label and a chance, represented by properties *hasNode*, *hasLabel* and *probability*, respectively. The *hasNode* property can reference only individuals of the *ChanceNode* class, since chance nodes are the only kind of node that represent random events. In order to specify a hard evidence (an observation of an event), we specialize the *Evidence* class creating the *HardEvidence* class. This class specifies a constraint defining that the *probability* property must assume the numeric value one.

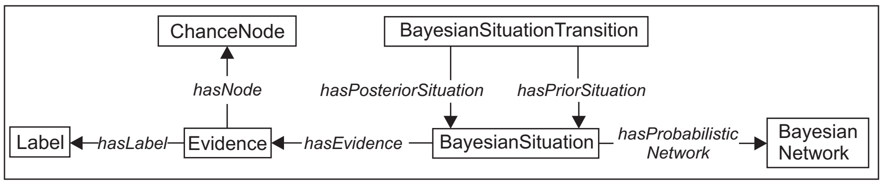


Fig. 4. Situation representation.

A situation, represented by the *Situation* class, has two properties. The first is the *hasProbabilisticNetwork* property used to reference the network individual whose configuration corresponds to the given situation. The second property is the *hasEvidence* that corresponds to the set of evidences that originates the situation. A particular kind of situation is represented by the *BayesianSituation* class. Its inherited *hasProbabilisticNetwork* property can reference only Bayesian networks.

In order to establish a link between two sequential situations we created a class named *SituationTransition*. This class is described by *hasPriorSituation* property and *hasPosteriorSituation* property, which represents a prior and a posterior situation, respectively. A situation transition between Bayesian networks is represented in the class *BayesianSituationTransition*. This class inherits the properties from *SituationTransition* and restricts them specifying that they can only reference *BayesianSituation* individuals.

4 Bayesian Agent Internal Architecture

The main goal of our agent internal architecture is to enhance the interoperability of Bayesian network knowledge among agents. The interoperability is achieved by an ontology-based approach to represent the uncertain knowledge of the agent. Following, we detail the agents' internal architecture, depicted in the Figure 5.

It is necessary to differentiate the architecture components from the agent implementation specific ones. The architecture components are represented in the figure by the gray elements. The *Agent Implementation Specific Components* are represented in the figure by the white element. They are not specified by this architecture since they relate to the particular purpose of each agent design. However, we specify the way they interact with the architecture components. Usually, the *Agent Implementation Specific Components* define the manner that the agent reasons about its goals and how it achieves them (i.e. planning and goal deliberation).

4.1 Architecture Components

The first component of the architecture is the *Perception Handler*, which receives and forwards the perceptions to the respective components capable of interpreting them. The characteristics of a perception (metadata) are taken into account to decide which component will receive it. Since in the context of this work we are dealing with interoperability among Bayesian agents, we focus on two particular categories of perception: *Bayesian Network Knowledge* and *Query*. The first corresponds to individuals of the ontology presented in the Section 3. The perceptions of this category are forwarded to the *Knowledge Base (KB) Update* component. The second corresponds to queries about the agent's knowledge that are forwarded to the *KB Query* component.

The second component of the architecture is the *KB Update*. Its purpose is to evaluate the incoming OWL *Bayesian Network Knowledge*, and insert the selected ones in the knowledge base as individuals of the Bayesian network ontology. The information to be inserted is selected following the criteria defined by the designer. A simple implementation of this component performs insertions

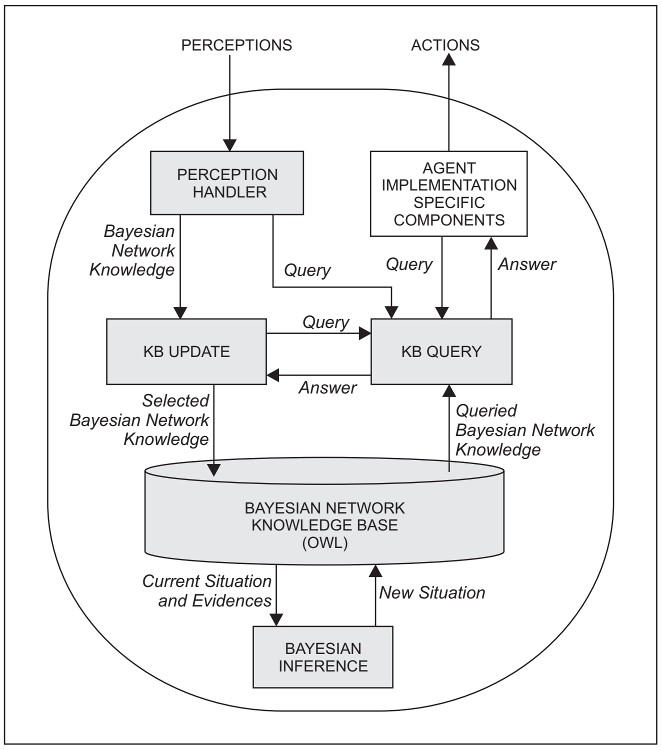

PERCEPTIONS
ACTIONS
PERCEPTION HANDLER
AGENT IMPLEMENTATION SPECIFIC COMPONENTS
Bayesian Network Knowledge
Query
Query
Answer
Query
KB UPDATE
KB QUERY
Answer
Selected Bayesian Network Knowledge
Queried Bayesian Network Knowledge
BAYESIAN NETWORK KNOWLEDGE BASE (OWL)
Current Situation and Evidences
New Situation
BAYESIAN INFERENCE

Fig. 5. Bayesian Agent Internal Architecture.

in the KB without restrictions. A more sophisticated implementation interacts with the *KB Query* component to retrieve already inserted Bayesian information to constrain the information to be inserted.

Our knowledge base is constituted by the Bayesian network ontology, detailed in the Section 3, and its individuals. It stores the Bayesian networks situations, the transitions between situations and the evidences. The base can contain multiple different Bayesian networks. Any modification in a Bayesian network characterizes a new situation, and the sequence of situations represents a history of a network. The history may be useful for an agent planning, in example.

In order to perform probabilistic reasoning in the Bayesian networks stored in the knowledge base, we specify the *Bayesian Inference* component. Its inputs are the *Current Situation* of a Bayesian network and a set of *Evidences*. The *Bayesian Inference* output is the *New Situation* with its probabilities recalculated considering the *Evidences*. It is worth to point out that both situations are individuals of the *BayesianNetwork* class and that the *Evidences* are *Evidence* class individuals. The *New Situation* resulting from the inference process constitutes the most up-to-date knowledge that the agent has about its domain. The presence of this component is indispensable since updated knowledge is necessary to support the agent decisions and actions.

The *KB Query* component receives queries from *Agent Implementation Specific Components*, *Perception Handler* and *KB Update*. These queries can return events (states) and their occurrence probabilities, causal relations between variables and other information that can be inferred from the Bayesian networks knowledge base. Queries from the agent specific components usually are performed to aid the agent in its decision making process. The queries forwarded by *Perception Handler* are related to knowledge that external agents need to be informed about. Finally, the queries from the *KB Update* component are executed with the purpose of selecting which information will be inserted on the KB.

The core of the interoperability relies on the Bayesian network ontology. It provides the fundamental domain concepts among the Bayesian agents making possible their knowledge exchange. The architecture supports the knowledge representation in a broader way, not only in the interaction with other Bayesian agents. The architecture also provides the means for the knowledge maintenance.

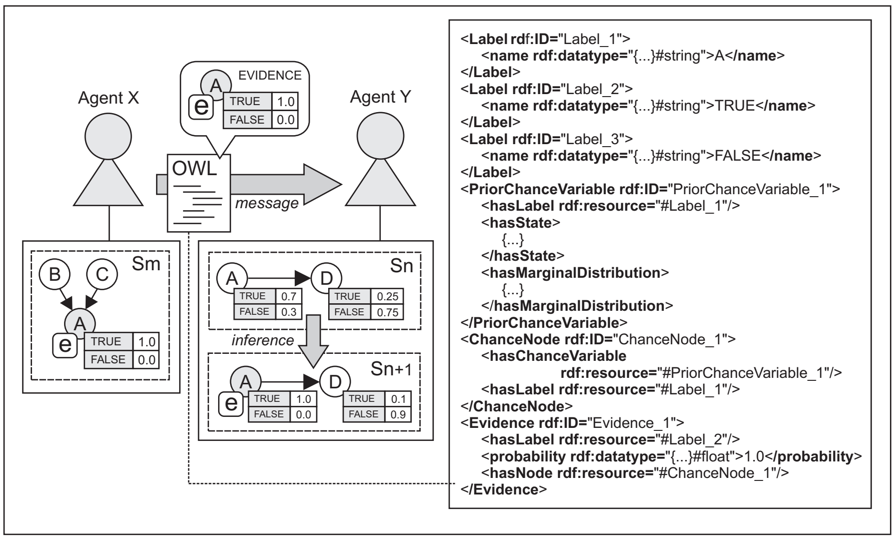
4.2 Interoperability Example

An example of interoperability between two Bayesian agents, *Agent X* and *Agent Y*, is illustrated in the Figure 6. Both agents have one Bayesian network in its knowledge base. Their Bayesian networks are different, but they have one node, labeled *A*, which represents the same information in both networks. In our approach, the *ChanceNode*, *ChanceVariable* and *State* classes have a property named *hasLabel*, which may indicate individuals that add semantics to the concepts represented by those Bayesian network elements.

The Bayesian network of *Agent X* has three nodes, labeled *A*, *B* and *C*. It is depicted only the current situation *Sm* of the network of the *Agent X*. In the situation *Sm*, the node *A* has an evidence that indicates the occurrence of the state *TRUE*. The Bayesian network of the *Agent Y* has two nodes, labeled *A* and *D*. It is shown the last two situations of the *Agent Y* network. The first, named *Sn*, represents the current situation before the execution of the inference process that considered the received evidence in the node *A*. The second is the actual situation, called *Sn+1*, resulting from the inference process. The inference is illustrated in the figure by a gray arrow from the situation *Sn* to the situation *Sn+1*.

A message exchanged among *Agent X* and *Agent Y* is represented by the gray arrow between them. The message, written in OWL language, contains the evidence associated with the node *A*. In the bottom we present a snippet of the OWL source code corresponding to the message content.

Following the flow of information, *Agent X* sends to *Agent Y* a message containing the evidence associated with the node *A*. Upon receiving the OWL message, the *Agent Y* updates its knowledge base by inserting the content of the message in it. Since the state of *A* is known, it is necessary to perform the inference to recalculate the probabilities associated with the node *D*. The



```
<Label rdf:ID="Label_1">
    <name rdf:datatype="{...}#string">A</name>
</Label>
<Label rdf:ID="Label_2">
    <name rdf:datatype="{...}#string">TRUE</name>
</Label>
<Label rdf:ID="Label_3">
    <name rdf:datatype="{...}#string">FALSE</name>
</Label>
<PriorChanceVariable rdf:ID="PriorChanceVariable_1">
    <hasLabel rdf:resource="#Label_1"/>
    <hasState>
        {...}
    </hasState>
    <hasMarginalDistribution>
        {...}
    </hasMarginalDistribution>
</PriorChanceVariable>
<ChanceNode rdf:ID="ChanceNode_1">
    <hasChanceVariable
                rdf:resource="#PriorChanceVariable_1"/>
    <hasLabel rdf:resource="#Label_1"/>
</ChanceNode>
<Evidence rdf:ID="Evidence_1">
    <hasLabel rdf:resource="#Label_2"/>
    <probability rdf:datatype="{...}#float">1.0</probability>
    <hasNode rdf:resource="#ChanceNode_1"/>
</Evidence>
```

Fig. 6. Bayesian Agents Interoperability Example.

inference generates a new situation *Sn+1* from the situation *Sn*, considering the evidence in the node *A*.

5 Case Study

The goal of this case study is to demonstrate a Bayesian knowledge exchange among the Social Agent and the Student Model Agent. They are Bayesian agents that belongs to PortEdu and AMPLIA [17] respectively. The idea is present a way to apply our agent architecture, allowing the Student Model Agent to send Bayesian information to the Social Agent [5].

PortEdu, a multiagent portal that hosts educational systems like Intelligent Tutoring Systems (ITS), provides infrastructure and services for the systems through an agent society. One of these agents is the Social Agent, responsible for organizing the users in groups considering cognitive and emotional aspects. The AMPLIA, one of the educational systems hosted in PortEdu, is an intelligent multiagent learning environment that focuses on the medical area. The functionalities of the AMPLIA are also provided by an agent society. The Student Model Agent, part of the AMPLIA multiagent system, represents the student beliefs in a specific domain and the confidence degree this learner has on the built network model.

The main objective of the Social Agent is to improve student's learning stimulating his interaction with other students, tutors and professors. The interaction is stimulated by recommending the students to join workgroups in order to provide and receive help from other students. The Social Agent's knowledge is implemented with Bayesian networks. In these networks it is represented student features such as social profile, acceptance degree, sociability degree, mood state, interest, commitment degree, leadership and performance. Figure 7 depicts the Bayesian network related to the student features. However, to communicate with PortEdu and AMPLIA agents, it is necessary to express such probabilistic knowledge in a way that these agents may process it. Such requirement is addressed using our agent architecture.

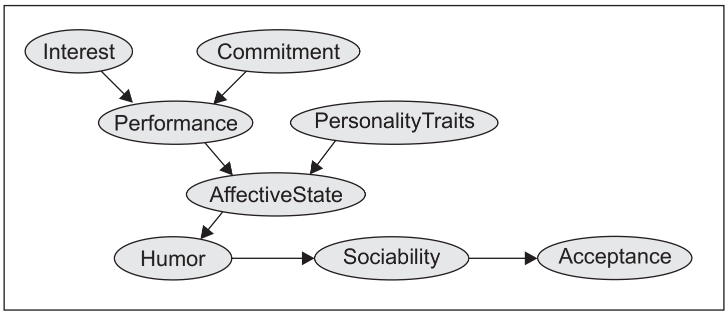


Fig. 7. Bayesian network representing student features.

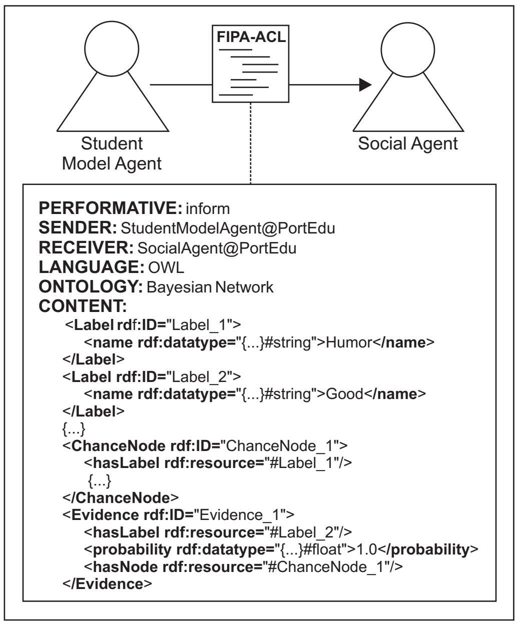
We begin the description of the architecture integration in the Social Agent specifying how the *Knowledge Base* component is implemented. The knowledge base is composed by the Bayesian network ontology specification (Section 3) and the ontology individuals (i.e. Bayesian networks, evidences, situation transitions). The network illustrated in the Figure 7, in example, is stored in the knowledge base of the Social Agent as a *BayesianNetwork* class individual. The ontology specification and initial population of the knowledge base were created in OWL using the Protégé tool [18].

The interaction of the Social Agent with other agents is done following the FIPA specifications, which are considered the current standard for interoperability among heterogeneous agents and, since 2005, are sponsored by IEEE. Considering the relevance of the FIPA standards, PortEdu adopts them for the platform specification and agent communication. The Social Agent was developed using the JADE [19] framework, which provides a FIPA-compliant middleware for multiagent system development. Developing an agent with this kind of abstraction allows more reutilization and directs the programming towards the agent-oriented paradigm.

We implemented the *Perception Handler*, *KB Update*, *KB Query* and *Bayesian Inference* as JADE Behaviors (implementations of agent's tasks) of the Social Agent. The *Perception Handler* manages interactions in compliance with FIPA protocols specifications, using JADE communication resources. In order to allow direct access to the *Knowledge Base* for the *KB Update* component we used the Jena [20] toolkit, which provides support for applications using OWL. Specifically, the current implementation of the *KB Update* component uses the Jena API to create and insert new individuals on the KB. The *KB Query* component also relies on Jena to execute the queries on the base. The *Bayesian Inference* component adopts the same algorithm used in the AMPLIA system. The inference is performed every time a new evidence is provided.

The actions of the Social Agent are considered in our architecture as *Agent Implementation Specific Components*. The main actions are the creation and management of the workgroups. Since our focus is on the proposed architecture components and its contributions to interoperability of Bayesian agents, we do not detail the decision making processes and action execution.

Since the architecture is implemented in the Social Agent, it is possible to perform an interaction aiming Bayesian knowledge exchange. The particular interaction defined in this section describes the interoperation of Bayesian evidence from Student Model Agent to Social Agent. The Figure 8 illustrates the Student Model Agent sending a FIPA-ACL message to Social Agent. The message performative is an inform, the content language is OWL and the agreed ontology specifies the Bayesian network domain.



```
PERFORMATIVE: inform
SENDER: StudentModelAgent@PortEdu
RECEIVER: SocialAgent@PortEdu
LANGUAGE: OWL
ONTOLOGY: Bayesian Network
CONTENT:
    <Label rdf:ID="Label_1">
        <name rdf:datatype="{...}#string">Humor</name>
    </Label>
    <Label rdf:ID="Label_2">
        <name rdf:datatype="{...}#string">Good</name>
    </Label>
    {...}
    <ChanceNode rdf:ID="ChanceNode_1">
        <hasLabel rdf:resource="#Label_1"/>
        {...}
    </ChanceNode>
    <Evidence rdf:ID="Evidence_1">
        <hasLabel rdf:resource="#Label_2"/>
        <probability rdf:datatype="{...}#float">1.0</probability>
        <hasNode rdf:resource="#ChanceNode_1"/>
    </Evidence>
```

Fig. 8. Interoperability among Social Agent and Student Model Agent.

In the message content is the OWL code of an *Evidence* individual that indicates the observation of the state *Good* in the node *Humor*. The reception of this evidence by the Social Agent will trigger the Bayesian inference process, generating a new situation in the Bayesian network illustrated in the Figure 7.

6 Conclusion and Future Work

In this paper, we present a way to interoperate Bayesian network knowledge among agents. In order to achieve it, we defined a Bayesian agent internal architecture (Section 4), an ontology to model the Bayesian network domain (Section 3), and developed a case study (Section 5) to demonstrate the integration of the architecture with the Social Agent, dealing with interoperability issues in the PortEdu environment.

Our approach to represent uncertain knowledge, differently from PR-OWL and PACL, does not propose any modification in standards like OWL or FIPA. We apply the current standards to provide a Bayesian knowledge representation through OWL. This approach allows our Bayesian agents to interoperate their knowledge and also contributes to researches on the expression of uncertain knowledge on the Semantic Web.

We define an internal architecture that provides support for knowledge intense agents to interoperate their knowledge. In this case, the interoperation is in the scope of Bayesian knowledge regarding an adequate way to express it. Besides that, the architecture provides resources for maintaining such Bayesian knowledge. Maintenance features allow the execution of updates, queries and an inference process to propagate evidences to the corresponding networks present in the knowledge base. The interoperability provided by this architecture aids agent specific decision making since it facilitates the discovery of new knowledge, allowing the agent to consider evidences that were not part of its original KB.

In our case study we concluded that our proposal can be integrated with the FIPA standards, more specifically with the FIPA-ACL. The adoption of OWL as a content language for ACL messages handles the issue of a common knowledge language. Our OWL ontology aggregates meaning to the message content. The utilization of OWL and the specification of the ontology to contextualize the content, allow the expression of knowledge in an open and explicit way.

Future works on ontology for Bayesian knowledge are twofold. The first is concerned with the executability aspects of the Bayesian networks. Its goal is to represent, in the ontology, concepts involved in the inference, expliciting this operational knowledge. This kind of knowledge allows an agent to share the way that the inference process is performed. The second corresponds to an extension of the ontology to describe also influence diagrams. In order to represent this kind of probabilistic network, decision and utility nodes must be incorporated in the ontology.

References

1. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an agent communication language. In: Proceedings of the 3rd International Conference on Information and Knowledge Management, Gaithersburg, MD, USA, ACM Press (1994) 456–463

2. The Foundation for Intelligent Physical Agents: Specifications. Available from http://www.fipa.org (2006)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
4. Nakayama, L., Vicari, R.M., Coelho, H.: An information retrieving service for distance learning. Transactions on Internet Research **1**(1) (2005) 49–56
5. Boff, E., Santos, E.R., Vicari, R.M.: Social agents to improve collaboration on an educational portal. In: IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society (2006) 896–900
6. Horrocks, I., Patel-Schneider, P., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics **1**(1) (2003) 7–26
7. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. Technical report, W3C (February 2004)
8. Ding, L., Kolari, P., Ding, Z., Avancha, S., Finin, T., Joshi, A.: Using Ontologies in the Semantic Web: A Survey. Technical report, UMBC (July 2005)
9. Ding, Z., Peng, Y.: A probabilistic extension to ontology language OWL. In: Hawaii International Conference On System Sciences. (2004)
10. da Costa, P.C.G., Laskey, K.B., Laskey, K.J.: PR-OWL: A bayesian ontology language for the semantic web. In: Workshop on Uncertainty Reasoning for the Semantic Web, International Semantic Web Conference. (2005) 23–33
11. Gasevic, D., Devedzic, V.: Petri net ontology. Knowledge Based Systems **19**(4) (2006) 220–234
12. Breton, E., Bézivin, J.: Towards an understanding of model executability. In: International Conference on Formal Ontology in Information Systems. (2001) 70–80
13. Gluz, J.C., Flores, C.D., Seixas, L., Vicari, R.M.: Formal analysis of a probabilistic knowledge communication framework. In: IBERAMIA/SBIA Joint Conference. (2006)
14. Santos, E.R., Boff, E., Vicari, R.M.: Semantic web technologies applied to interoperability on an educational portal. In: Intelligent Tutoring Systems, 8th International Conference. Volume 4053., Springer (2006) 308–317
15. Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J.: Probabilistic Networks and Expert Systems. Springer Verlag (1999)
16. Pearl, J.: Belief networks revisited. Artificial Intelligence **59**(1-2) (1993) 49–56
17. Vicari, R.M., Flores, C.D., Silvestre, A.M., Seixas, L.J., Ladeira, M., Coelho, H.: A multi-agent intelligent environment for medical knowledge. Artificial Intelligence in Medicine **27**(3) (2003) 335–366
18. Stanford University: The Protégé Ontology Editor and Knowledge Acquisition System. Available from http://protege.stanford.edu
19. Bellifemine, F., Poggi, A., Rimassa, G.: JADE – A FIPA-compliant agent framework. In: Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology. (1999) 97–108
20. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the semantic web recommendations. Technical report, Hewlett Packard Laboratories (December 2003)